NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

MITIGATING DISTRIBUTED DENIAL OF SERVICE ATTACKS WITH MULTI-PROTOCOL LABEL SWITCHING —TRAFFIC ENGINEERING (MPLS-TE)

by

Ioannis Vordos

March 2009

Thesis Advisor:	Geoffrey Xie
Second Reader:	John D. Fulp

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE	Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	March 2009	Master's Thesis

4. TITLE AND SUBTITLE: Mitigating Distributed Denial of Service Attacks with Multi-Protocol Label Switching—Traffic Engineering (MPLS-TE)	5. FUNDING NUMBERS
6. AUTHOR(S) Ioannis Vordos	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A	10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT	12b. DISTRIBUTION CODE
Approved for public release; distribution is unlimited	

13. ABSTRACT (maximum 200 words)

A Denial of Service (DoS) occurs when legitimate users are prevented from using a service over a computer network. A Distributed Denial of Service (DDoS) attack is a more serious form of DoS in which an attacker uses the combined power of many hosts to flood and exhaust the networking or computing resources of a target server. In recent years, DDoS attacks have become a major threat to both civilian and military networks.

Multi-Protocol Label Switching with Traffic Engineering (MPLS-TE) is an emerging technology that allows explicit, bandwidth-guaranteed packet forwarding paths to be established for different traffic flows. It provides a means for diverting packets of a suspected DDoS attack for analysis and cleaning before forwarding them to the actual destination.

The objective of this research was to implement and evaluate the performance of an MPLS-TE based solution against DDoS attacks on a realistic test-bed network consisting of Cisco routers. The test-bed has been integrated with Snort®, an open source Intrusion Detection System (IDS), to achieve automatic detection and to mitigate DDoS attacks. The test-bed network was subject to a series of malicious traffic flows with varying degrees of intensity. The results demonstrated that MPLS-TE is very effective in mitigating such attacks. The overall system response time and the router CPU loads are comparable to those reported by two former NPS theses that examined alternative solutions based on BGP blackhole routing.

14. SUBJECT TERMS Traffic Engineering, Distributed Denial of Service Attacks, Sinkhole Routing, Blackhole Routing	15. NUMBER OF PAGES 135
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UU

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

MITIGATING DISTRIBUTED DENIAL OF SERVICE ATTACKS WITH MULTI-PROTOCOL LABEL SWITCHING—TRAFFIC ENGINEERING (MPLS-TE)

Ioannis Vordos
Lieutenant, Hellenic Navy
B.S., Hellenic Naval Academy, 1994

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
March 2009

Author: Ioannis Vordos

Approved by: Geoffry Xie, Ph.D.
Thesis Advisor

John D. Fulp, Senior Lecturer
Second Reader

Peter J. Denning, Ph.D.
Chairman, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

A Denial of Service (DoS) occurs when legitimate users are prevented from using a service over a computer network. A Distributed Denial of Service (DDoS) attack is a more serious form of DoS in which an attacker uses the combined power of many hosts to flood and exhaust the networking or computing resources of a target server. In recent years, DDoS attacks have become a major threat to both civilian and military networks.

Multi-Protocol Label Switching with Traffic Engineering (MPLS-TE) is an emerging technology that allows explicit, bandwidth-guaranteed packet forwarding paths to be established for different traffic flows. It provides a means for diverting packets of a suspected DDoS attack for analysis and cleaning before forwarding them to the actual destination.

The objective of this research was to implement and evaluate the performance of an MPLS-TE based solution against DDoS attacks on a realistic test-bed network consisting of Cisco routers. The test-bed has been integrated with Snort®, an open source Intrusion Detection System (IDS), to achieve automatic detection and to mitigate DDoS attacks. The test-bed network was subject to a series of malicious traffic flows with varying degrees of intensity. The results demonstrated that MPLS-TE is very effective in mitigating such attacks. The overall system response time and the router CPU loads are comparable to those reported by two former NPS theses that examined alternative solutions based on BGP blackhole routing.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

First, I thank the Hellenic Navy for providing the opportunity to pursue my studies at the Naval Postgraduate School. It was a dream for me since many years back.

I would like to thank my thesis advisor professor Geoffrey Xie for the opportunity he gave me to work on such an interesting and challenging thesis topic. He provided me very generously all the required knowledge and means to achieve my goal.

I would also like to thank all the rest of the Academic Staff of the Naval Postgraduate School and especially the Department of Computer Science for the qualitative knowledge that they accorded me with a high sense of responsibility.

Special thanks to my thesis editor, Mrs. Barbara Young. She helped me to improve my writing and her advices made my work easier.

Last and more important, I would like to thank my family, my wife, Natassa, and my two daughters, Andriana and Stella. Without their unconditional and absolute support, my goals would have never been succeeded.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

Denial of Service (DoS) is a common type of cyberattack over the Internet. The purpose of DoS is to make a computer's resources unavailable to its intended users. One way to launch a DoS attack is by sending malformed traffic to the target or by sending a huge amount of normal traffic which will overload the target's buffer. To be more effective, attackers often use many compromised machines, rather than just one, as a source for the attack. In such a case, the malicious packets approach the victim from different locations. This special type of DoS, called Distributed Denial of Service (DDoS), is one of the most difficult problems affecting normal operations on the Internet.

The first well-documented DDoS attack occurred in August 1999, when a DDoS tool called Trinoo was deployed and activated in at least 227 hosts, flooding a single University of Minnesota computer. That computer was down for more than two days as a result [1].

The biggest DDoS attack in terms of duration, number of victims, and caused damage started on February 7, 2000. Yahoo! was one of the first victims and the Internet portal was inaccessible by users for three hours. Analysts estimated that due to this attack Yahoo suffered a loss of e-commerce and advertising revenue amounting to about $500,000. On the same day, CNN, eBay, Amazon and Buy.com, were all victims of DDoS attacks, causing them to either stop functioning completely or slowing their response times down significantly. According to book seller Amazon.com, the attacks resulted in a loss of $600,000 during the 10 hours its Web site was down. Buy.com went from 100% availability to 9.4%, while CNN.com's users went down to below 5% of normal volume. And, on February 9, E*Trade and ZDNet both suffered DDoS attacks. E*Trade was virtually unreachable. One can only assume that to a company that does $2 billion dollars weekly in online trades, the downtime loss was huge.

Such DDoS attacks are a major concern to the military. A continuous flow of information is critical to modern military operations. Additionally, military networks are increasingly based on the same technologies used by the public Internet, making them susceptible to the same wide range of DDoS threats.

Several techniques exist to protect a network's hosts against a DDoS attack by filtering out malicious packets. One of the most common is the "Border Gateway Protocol (BGP) Blackhole Routing."

Blackhole routing (BHR) is a clever way of implementing the policy "route this packet to the trash." The concept is quite simple and leverages the basic operation of routers. A blackhole route tells the router to send the suspected packets to the *null0* interface (a non-existent interface), which is equivalent to telling the router to "route this packet to the trash."

In prior efforts, two former NPS students [2] [3] built test-beds to investigate how to mitigate DDoS with BGP blackhole routes. In the first one (Stamatelatos' Master's thesis), the author evaluated the performance of BHR methods in the lab with three real-time test-bed networks which were manually triggered by the administrator.

The second one (Puri's Master's thesis) used the results from Stamatelatos' study in combination with a proper IDS system and the result was a working implementation of a fully automated attack-detect react-protect BHR system.

The problem with BHR is that it protects only the network, not the victim. It directs all the traffic - good or bad - to the "trash" and the target cannot receive any traffic during the attack. So, good traffic is also sent to the "trash" and thus, the DDoS has still achieved its purpose of DoS'ing the target machine.

This thesis will evaluate another more recently developed technique for DDoS mitigation. It is based on MultiProtocol Label Switching – Traffic Engineering (MPLS –TE).

The MPLS is a connection-oriented forwarding mechanism in which packets are forwarded based on labels. It was introduced in 2001 in an attempt to create a faster forwarding mechanism to combine the advantages of the already existing Internet Protocol (IP) and the Asynchronous Transfer Mode (ATM).

Traffic engineering refers to a mode of network operation whereby routes are selected specifically to meet the delay and throughput requirements of individual user traffic flows. The MPLS's support for explicit routing allows network engineers to adjust the routing of flows to balance the use of a network's resources and implement traffic engineering solutions. So, MPLS traffic engineering (MPLS-TE) provides a way to achieve traffic engineering benefits without needing to run a separate network and without needing a non-scalable full mesh of router interconnects.

With MPLS-TE, when an attack is occurring, all traffic destined to the victim can be redirected; not sent to the "trash", but rather to a Cleaning Center connected on one of the Label Edge Routers (LERs), as border routers are called in MPLS terminology. In this Cleaning Center the traffic will be analyzed and "cleaned"; i.e., malicious traffic is sent to the "trash" and the good traffic is redirected back to its original destination–the attack's target.

The research questions that will be answered by the research for this thesis are the following:

1. What is MultiProtocol Label Switching (MPLS)? What are the goals of MPLS?

2. How is Traffic Engineering implemented with MPLS?

3. What are common types of DDoS attacks that can be mitigated by the MPLS-TE techniques?

4. What is the difference between the MPLS-TE technique and the BGP Blackholing?

5. What is the speed of reaction of the proposed MPLS-TE technique to a new DDoS attack?

The rest of the thesis is organized as follows. Chapter II presents a more detailed explanation of DDoS attacks, MPLS-TE, previous studies of DDoS mitigation with BGP Blackhole routing (BHR), methods, and the already existing techniques for DDoS mitigation with MPLS-TE. Chapter III describes the methodology and the test-bed network configuration used in this research. Chapter IV presents the results and analysis of this research and a comparison between MPLS-TE and BGP BHR methods. Chapter V provides conclusions and suggestions for future work.

II. BACKGROUND

A. CHAPTER OVERVIEW

This chapter provides background information for this study. The first section describes the basic attributes of Distributed Denial of Service (DDoS) attacks and the most common techniques that attackers use. The second section presents the principles of MultiProtocol Label Switching – Traffic Engineering (MPLS-TE) forwarding technique. The third section presents the results from previous studies related to Boarder Gateway Protocol (BGP) Blackhole routing for DDoS attack's mitigation The fourth section describes the existing methods of implementation that have been proposed to protect a network from DDoS attacks with the MPS-TE technique.

B. DISTRIBUTED/DE NIAL OF SERVICE ATTACKS

A Denial of Service (DoS) attack can be characterized as an attack on a server or a network with the purpose of preventing legitimate users from using that server or network. A Distributed Denial of Service (DDoS) attack is a large-scale coordinated DoS attack on the availability of services of a server or network, launched indirectly, through many compromised computers on the Internet. The services under attack are those of the "primary victim," while the compromised systems used to launch the attack are often called "Zombies" or "secondary victims." The use of secondary victims in performing a DDoS attack provides the attacker with the ability to perform a much larger and more disruptive attack, while making it more difficult to track down the original attacker. As defined by the World Wide Web Security FAQ:

> A Distributed Denial of Service (DDoS) attack uses many computers to launch a coordinated DoS attack against one or more targets. Using client/server technology, the perpetrator is able to

multiply the effectiveness of the Denial of Service significantly by harnessing the resources of multiple unwitting accomplice computers which serve as attack platforms [4].

There are many kinds of DDoS attacks. In general we can divide them into four main classes based on how they are engineered: Flood Attacks, Amplifications Attacks, Protocol Exploit Attacks and Malformed Packets Attacks [5].

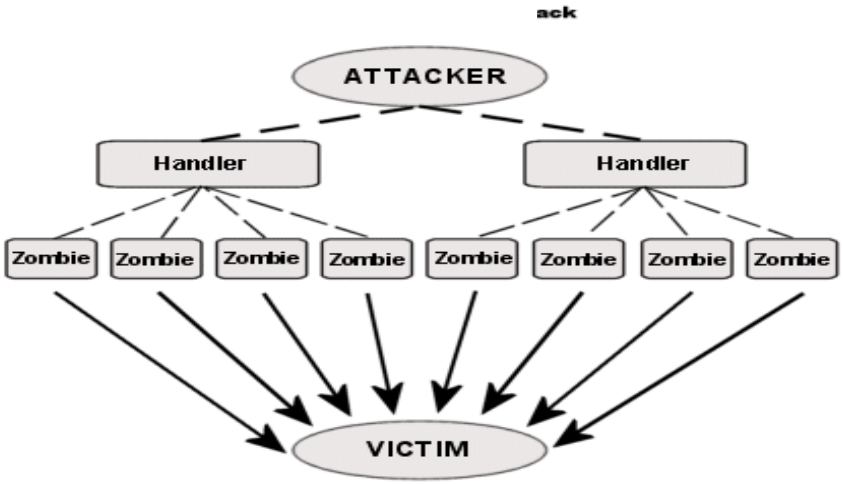


Figure 1. Architecture of a DDoS Attack (From: [6])

1. Flood Attacks

In Flood Attacks, the attacker uses the Zombies to send large amounts of traffic to the victim's system, in order to congest the victim system's network bandwidth with IP traffic. The system under attack slows down, crashes, or suffers, or denies access to legitimate users. Flood attacks can be launched using both User Datagram Protocol (UDP) and Internet Control Message Protocol (ICMP) packets.[5]

a. UDP Flood Attack

In a UDP Flood attack, the attacker sends a large number of UDP packets through the Zombies to either random or specified ports on the victim's

system. Often, the attacking DDoS tool will also spoof the source IP address of the attacking packets. This helps hide the identity of the secondary victims since return packets from the victim's system are not sent back to the Zombies, but to the spoofed addresses.

The victim's system tries to process the incoming data to determine which applications have requested data. If the victim's system is not running any applications on the targeted port, it will send out an ICMP packet to the sending system indicating a "destination port unreachable" message.

Thus, for a large number of UDP packets, the victimized system will be forced into sending many ICMP packets, eventually leading it to be unreachable by other clients. A UDP flood attack may also fill the bandwidth of connections located around the victim's system. This often impacts systems located near the victim.[5]

b. ICMP Flood Attack

In ICMP flood attacks, the attacker sends a large number of ICMP_ECHO packets ("ping") to the victim's system through the Zombies. These packets cause the victim's system to reply. The combination of inbound and outbound traffic saturates the bandwidth of the victim's network connection [5]. Often, the attacking DDoS tool will also spoof the source IP address of the attacking packets. This helps hide the identity of the secondary victims since return packets from the victim system are not sent back to the Zombies, but to the spoofed addresses [5].

Due to its simplicity this kind of attack is the chosen attack to be contacted during this thesis' testing. One more reason making this kind of attack desirable for examination is that it has been used in previous studies with BGP BHR techniques. Since this thesis is going to compare this current technique's performance with the earlier technique, it is very important for both techniques to at least be contacted under the same kind of DDoS attack.

2. Amplification Attacks

In amplification attacks the attacker spoofs the target's IP address and he or the Zombies send messages to a broadcast IP address, trying to cause all systems in the subnet reached by the broadcast address to send a reply to the victim's system. Most routers have the broadcast IP address feature. When a sending system specifies a broadcast IP address as the destination address, the routers replicate the packet and send it to all the IP addresses within the broadcast address range. That is where the attack's name comes from. The broadcast IP address is used to amplify and reflect the attack traffic, and thus reduce the victim system's bandwidth [5].

The attacker can send the broadcast message directly, or use the Zombies to send the broadcast message to increase the volume of attacking traffic. If the attacker decides to send the broadcast message directly, this attack provides the attacker with the ability to use the systems within the broadcast network as Zombies without needing to gain access to them or to install any agent software [5].

a. Smurf Attack

The Smurf attack is named after the source code employed to launch the attack (smurf.c) [7]. A Smurf attack uses ICMP_ECHO_REQUEST packets with a spoofed source address of the victim. The destination of those packets is an IP network broadcast address. When the systems on the network (amplifiers) where the broadcast address, the ECHO_REQUEST is sent receive the packet with the falsified source address (i.e., the return address), they respond, flooding the targeted victim with the echo replies. The amplifier sends the ICMP ECHO REQUEST packets to all of the systems within the broadcast address range, and each of these systems will return an ICMP ECHO REPLY to the target victim's IP address. This flood can overwhelm the targeted victim's network. Both the intermediate and victim's networks will see degraded

performance. The attack can eventually result in the inoperability of both networks. This type of attack amplifies the original packet tens or hundreds of times.

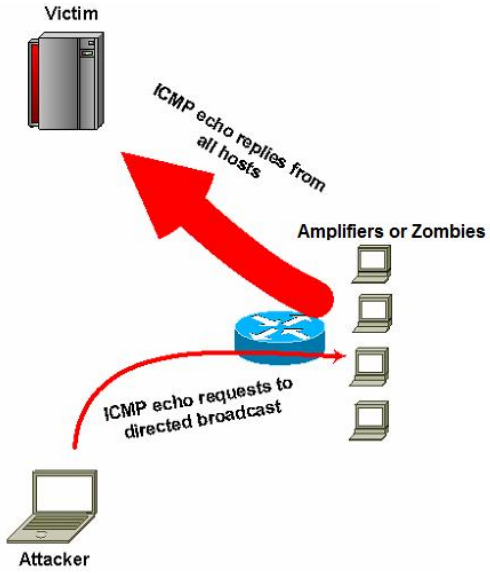


Figure 2. An example of a Smurf attack (After: [2])

b. Fraggle Attack

Another example of amplification attacks is a DDoS Fraggle attack, where the attacker sends packets to a network amplifier, using UDP ECHO packets instead of ICMP ECHO packets used in Smurf attacks. The result is almost the same as with the Smurf attacks [5].

3. Protocol Exploit Attacks

This category of DDoS attacks is based upon IP protocol's vulnerabilities. Two examples are given below. The first one is about misuse of the TCP SYN (Transfer Control Protocol Synchronize) protocol, and the second one about the misuse of the PUSH+ACK protocol [5].

a. TCP SYN Attack

The TCP SYN attack exploits the three-way handshake between the sender and the receiver by sending a large amount of TCP SYN packets to the victim's system with spoofed source IP addresses, so the victim system responds with a SYN+ACK packet to each of them. When the received malformed SYN requests are being processed by a server and none of the ACK responses are returned, the server eventually runs out of processor and memory resources, and becomes unable to respond to legitimate users. Basically, SYN flooding disables a targeted system by creating many half-open connections. Each operating system has a limit to the number of connections it can accept. In addition, the SYN flood may exhaust system memory, resulting in a system crash. In a DDoS TCP SYN attack, the attacker uses Zombies to send large amount of bogus TCP SYN requests to the victim's server in order to reserve the server's processor resources, and hence prevent the server from responding to legitimate requests.

b. Push + Ack Attack

The PUSH + ACK attack is similar to a TCP SYN attack regarding its purpose that is to reduce the resources of the victim's system. In a PUSH + ACK attack, the attacker, through the Zombies, sends TCP packets with the PUSH and ACK flags (bits) set to one. These flags in the TCP header instruct the victim system to empty all data in the TCP buffer (regardless what the buffer contains) and send an acknowledgement when complete. If this sequence is repeated with multiple Zombies, the receiver cannot process the large amount of incoming packets and the victim's system will run out of resources [8].

4. Malformed Packet Attacks

As S. M. Specht and R. B. Lee stated in their paper:

A malformed packet attack is an attack where the attacker instructs the zombies to send incorrectly formed IP packets to the victim system in order to crash it.

There are a variety of malformed packet attacks. The most known [9] are:

Land Attack, Latierra Attack, Ping of Death Attack, Jolt2 Attack, Rose Attack, Teardrop, Newtear, Bonk, Syndrop Attack, and Winnuke Attack.

C. MULTIPROTOCOL LABEL SWITCHING (MPLS) – TRAFFI C ENGINEERING (TE)

1. What is MPLS

In accordance with IEC's site:

Multiprotocol label switching (MPLS) is a versatile solution to address the problems faced by present-day networks – speed, scalability, quality-of-service (QoS) management, and traffic engineering. MPLS has emerged as an elegant solution to meet the bandwidth-management and service requirements for next-generation Internet protocol (IP)–based backbone networks. MPLS addresses issues related to scalability and routing (based on QoS and service quality metrics) and can exist over existing asynchronous transfer mode (ATM) and frame-relay networks [10].

With an IP forwarding mechanism, packets are sent from a source to a destination in a hop-by-hop manner. Intermediate routers examine each packet's header and perform a route table lookup to determine the next hop (i.e., router) toward the destination. This may consume a network's resources because of the increased CPU requirements to process each packet's header. Although modern routers use hardware and software switching techniques to manage the headers' examination process by creating high-speed cache entries, these methods rely upon the Layer 3 routing protocol to establish the path to the destination.

The problem with this approach is that routing protocols have little knowledge about Layer's 2 characteristics, such as loading and quality of service

(QoS). Continuously increased demand for higher quantity and better quality of traffic puts demanding pressure on the Internet's backbone.

To meet these new demands, multiprotocol label switching (MPLS) abandoned the hop-by-hop technique by enabling devices to specify paths in the network based upon QoS and bandwidth needs of the applications. In other words, route selection can now take into account Layer 2's attributes. Before MPLS, vendors implemented other techniques for switching frames with values other than the Layer 3 header.

In 2001, based on Cisco's tag-switching protocol, the IETF defined MPLS as a vendor-independent protocol. Although the two protocols have much in common, differences between them prevent tag-switching devices from interacting directly with MPLS devices. MPLS has now superseded tag switching [11].

2. How MPLS Works

In accordance with Cisco's Principal Consultant, Cisco Systems-India & SAARC, Chandan Mendiratta

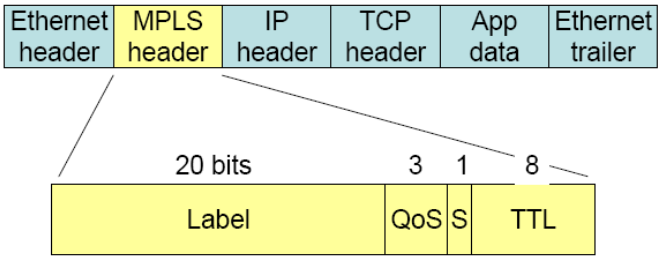
> MPLS is a scheme typically used to enhance an IP network. Routers on the incoming edge of the MPLS network add an 'MPLS label' to the top of each packet. This label is based on some criteria (e.g. destination IP address) and is then used to steer it through the subsequent routers. The routers on the outgoing edge strip it off before final delivery of the original packet. MPLS can be used for various benefits such as multiple types of traffic coexisting on the same network, ease of traffic management, faster restoration after a failure, and, potentially, higher performance. [12]

So, the main idea is to add a small label (sometimes called a "tag") on the front of a packet and route the packet based on the label, instead of the IP address. The MPLS operates at an OSI Model layer that lies between traditional definitions of Layer 2 (Data Link Layer) and Layer 3 (Network Layer), and therefore is often called the "Layer 2.5" protocol [13]. It provides data-carrying service for both circuit-based clients and packet-switching clients which provide a

datagram service model. It can be used to carry many different kinds of traffic, including IP packets, as well as ATM and Ethernet frames.

In order to further understand how this protocol works, it is critical for the reader to be familiar with the following definitions:

• **Label**—A header created by an edge label switch router (edge LSR) and used by label switch routers (LSR) to forward packets. The header format varies based upon the network media type. For example, in an ATM network, the label is placed in the VPI/VCI fields of each ATM cell header. In a LAN environment, the header is a "shim" located between the Layer 2 and Layer 3 headers. This thesis research is concerned only with IP packets and labels



Label : Label value (0 to 15 are reserved for special use)
QoS : Quality of Service
S : Bottom of Stack (set to 1 for the last entry in the label)
TTL : Time To Live

Figure 3. Shim headers are used for most non-ATM networks (From: [14])

• **Label Switch Router (LSR)**—A device such as a switch or a router that forwards labeled entities based upon the label's value.

• **Label Edge Router (LER)** —Resides at the edge of an MPLS network and assigns and removes the labels from the packets.

• **Label Sw itched**—When an LSR makes a forwarding decision based upon the presence of a label in the frame/cell.

• **Label-Switched Path (LSP)** —The path defined by the labels through LSRs between end points.

• **Forward Equivalence Class (FEC)** – A representation of a group of packets that share the same requirements for their transport. The assignment of a particular packet to a particular FEC is done just once (when the packet enters the network).

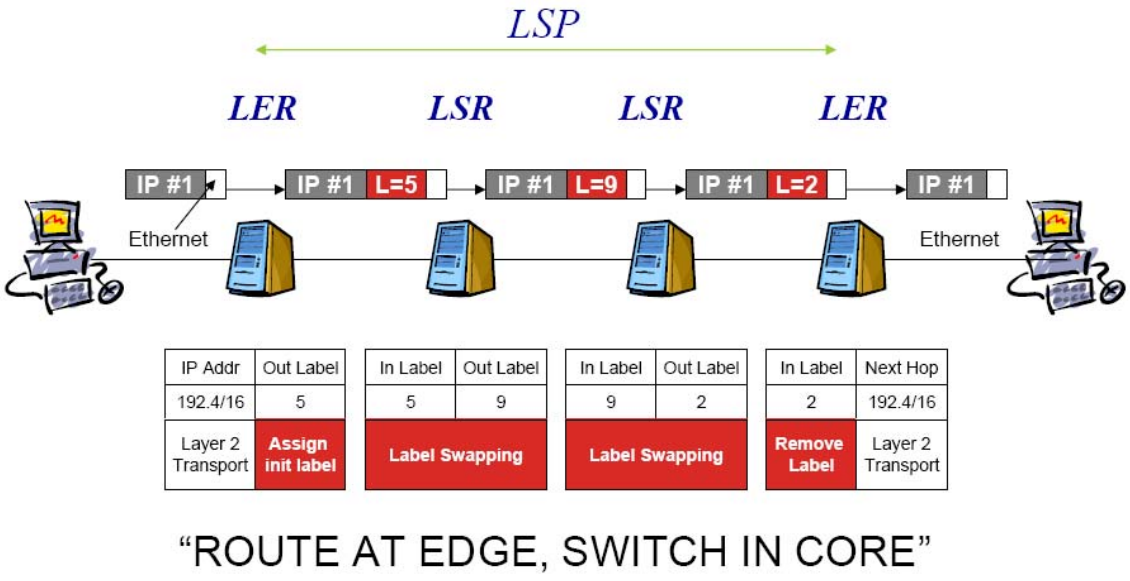


Figure 4. Basics about MPLS (From: [15])

As stated on Juniper's corresponding page "How MPLS Works" [16]:

MPLS is not a routing protocol; it works with layer 3 routing protocols (BGP, IS-IS, OSPF) to integrate network layer routing with label switching. An MPLS FEC consists of a set of packets that are all forwarded in the same manner by a given label-switching router (LSR). For example, all packets received on a particular interface might be assigned to a FEC. MPLS assigns each packet to a FEC only at the LSR that serves as the ingress node to the MPLS domain. A label distribution protocol binds a label to the FEC. Each LSR uses the label distribution protocol to signal its forwarding peers and distribute its labels to establish an LSP. The label distribution protocol enables negotiation with the downstream LSRs to determine what labels are used on the LSP and how they are employed.

> Labels represent the FEC along the LSP from the ingress node to the egress node. The label is prepended to the packet when the packet is forwarded to the next hop. Each label is valid only between a pair of LSRs. A downstream LSR reached by a packet uses the label as an index into a table that contains both the next hop and a different label to prepend to the packet before forwarding.
>
> The above section closes as follows [16]:
>
> The LSR that serves as the egress MPLS node uses the label as an index into a table that has the information necessary to forward the packet from the MPLS domain. The forwarding actions at the egress LSR can be any of the following:
>
> Forward the packet based on the inner header exposed after popping the label. This can be accomplished either by doing a routing table lookup or forwarding based on the exposed inner MPLS label.
>
> Forward the packet to a particular neighbor as directed by the table entry, for example in a Martini layer 2 transport case.

Each LSR, also known as an MPLS node, must support the following [16].

- At least one Layer 3 routing protocol (IS-IS, OSPF or BGP)
- A label distribution protocol (LDP, BGP, or RSVP-TE)
- The ability to forward packets based on their labels

An LSP with MPLS can be defined either by hop-by-hop routing (where each LSR independently selects the next hop for a given FEC), or by explicit routing (similar to source routing – the ingress LSR specifies the list of nodes through which the packet traverses (Traffic Engineering). The LSP setup for an FEC is unidirectional. The return traffic must use another LSP (may be the same if defined so) [16].

When an MPLS network has been set up, the routing protocol (OSPF, BGP or IS-IS) is used to specify how routers can communicate with each other with the routing update messages.

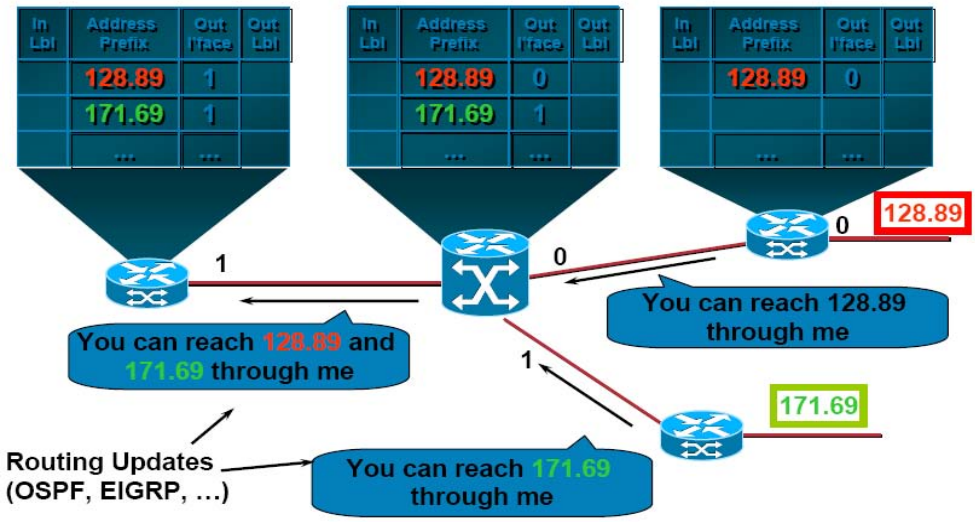


Figure 5. A MPLS network example: Exchange routing information (From: [17])

Sequentially, the selected label distribution protocol (LDP, BGP or RSVP-TE) is used to assign the corresponding labels.

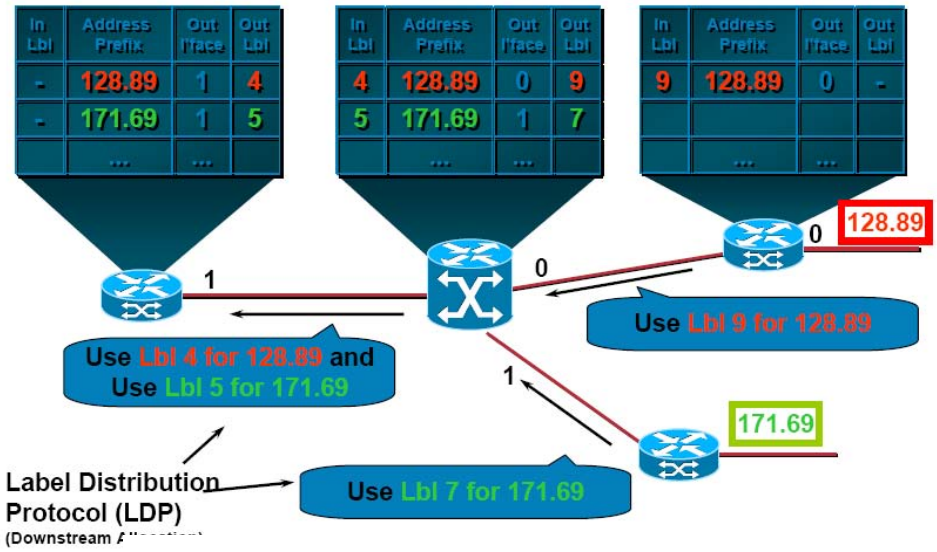


Figure 6. A MPLS network example: Assigning Labels (From: [17])

After the label assignment, the MPLS network is ready to forward packets as already described.

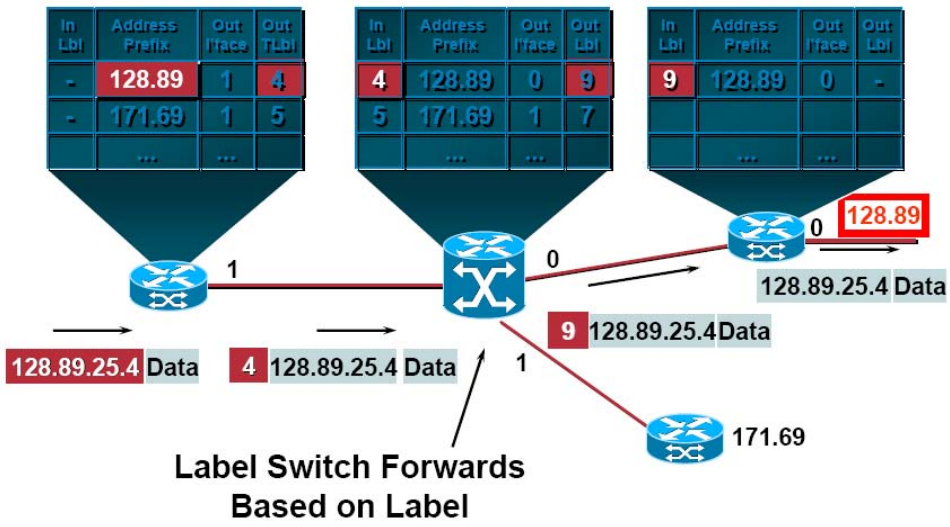


Figure 7. A MPLS network example: Forwarding packets (From: [17])

3. Label Distribution

There are three methods for label distribution. The first one is the Label Distribution Protocol (LDP). This LDP is used between nodes in an MPLS network to establish and maintain the label bindings. In order for MPLS to operate correctly, label distribution information needs to be transmitted reliably, and the label distribution protocol messages pertaining to a particular FEC need to be transmitted in sequence. Flow control is also desirable, as is the capability to carry multiple label messages in a single datagram.

As described on protocols.com “MPLS” web page [18], the LSR uses LDP in order

> …to establish label switched paths through a network by mapping network layer routing information directly to data-link layer switched paths. These LSPs may have an endpoint at a directly attached neighbor (like IP hop-by-hop forwarding), or may have an endpoint at a network egress node, enabling switching via all intermediary nodes. A FEC (Forwarding Equivalence Class) is associated with each LSP created. This FEC specifies which packets are mapped to that LSP.

The second method is with RSVP, which is used in MPLS traffic engineering. This method employs additions to the RSVP signaling protocol. It leverages the admission control mechanism of RSVP. Label requests are sent in PATH messages and binding is done with RESV messages. An EXPLICIT-ROUTE object defines the path over which setup messages should be routed. Using RSVP has several advantages [17].

The advantages of using RSVP with MPLS and how it works are very well described in protocols.com web page [18] as follows:

> The RSVP protocol defines a session as a data flow with a particular destination and transport-layer protocol. However, when RSVP and MPLS are combined, a flow or session can be defined with greater flexibility and generality. The ingress node of an LSP (Label Switched Path) uses a number of methods to determine which packets are assigned a particular label. Once a label is assigned to a set of packets, the label effectively defines the flow through the LSP. We refer to such an LSP as an LSP tunnel because the traffic through it is opaque to intermediate nodes along the label switched path.

The last method for label distribution is the BGP-Based Label Distribution, which is used in the context of MPLS VPNs. Since VPNs have nothing to do with this research effort this last method is not addressed further in this thesis.

4. What is MPLS-TE

In accordance with Wikipedia, Teletraffic or Traffic Engineering is:

> …the application of traffic engineering theory to telecommunications. Teletraffic engineers use their basic knowledge of statistics including Queueing theory, the nature of traffic, their practical models, their measurements and simulations to make predictions and to plan telecommunication networks at minimum total cost. These tools and basic knowledge help provide reliable service at lower cost. [19]

The MPLS-TE software allows an MPLS backbone to simulate and expand upon the traffic engineering capabilities of Layer 2 Frame Relay networks and ATM [20].

As is referred to on Cisco's MPLS-TE web page [20]:

> Traffic engineering is essential for service provider and Internet service provider (ISP) backbones. Such backbones must support a high use of transmission capacity, and the networks must be very resilient, so that they can withstand link or node failures.
>
> MPLS traffic engineering provides an integrated approach to traffic engineering. With MPLS, traffic engineering capabilities are integrated into Layer 3, which optimizes the routing of IP traffic, given the constraints imposed by backbone capacity and topology.
>
> MPLS traffic engineering routes traffic flows across a network based on the resources the traffic flow requires and the resources available in the network.
>
> MPLS traffic engineering employs "constraint-based routing," in which the path for a traffic flow is the shortest path that meets the resource requirements (constraints) of the traffic flow. In MPLS traffic engineering, the flow has bandwidth requirements, media requirements, a priority versus other flows, and so on.
>
> MPLS traffic engineering gracefully recovers to link or node failures that change the topology of the backbone by adapting to the new set of constraints.

5. How MPLS-TE Works

As has already been discussed, MPLS can be considered an integration of Layer 2 and Layer 3 technologies. The MPLS enables traffic engineering by making traditional Layer 2 features available (or "visible") to Layer 3. Thus, vendors can provide in a one-tier network that traditional techniques could only achieve by overlaying a Layer 3 network on a Layer 2 network [20].

As stated on Cisco's corresponding web page [20]:

> MPLS traffic engineering automatically establishes and maintains the tunnel across the backbone, using RSVP. The path used by a given tunnel at any point in time is determined based on the tunnel resource requirements and network resources, such as bandwidth. Available resources are flooded via extensions to a link-state based Interior Protocol Gateway (IPG). Tunnel paths are calculated at the tunnel head based on a fit between required and available resources (constraint-based routing). The IGP automatically routes the traffic into these tunnels. Typically, a packet crossing the MPLS traffic engineering backbone travels on a single tunnel that connects the ingress point to the egress point.

A tunnel is a path that can either be:

- explicitly configured hop-by-hop,
- dynamically routed by the Constrained Shortest Path First (CSPF) algorithm, or
- configured as a loose route that avoids a particular IP or that is partly explicit and partly dynamic.

In order to achieve MPLS-TE, the engaged routers should support the following mechanisms, as they are defined on Cisco's site [20]:

- Label-switched path (LSP) tunnels, are signaled through RSVP, with traffic engineering extensions. The LSP tunnels are represented as tunnel interfaces. Tunnels have a preconfigured destination, and they are unidirectional. This last issue means that a return tunnel must be established if full duplex communication is desired.
- A link-state IGP (such as OSPF) with extensions for the global flooding of resource information, and extensions for the automatic routing of traffic onto LSP tunnels must be selected as appropriate.
- An MPLS-TE path calculation module determines paths to use for LSP tunnels. This is not necessary if the tunnel configuration is manually created, such as in a LAN or small WAN.

- An MPLS-TE link management module that does link admission and bookkeeping of the resource information to be flooded.

- Label switching forwarding, provides routers with a Layer 2-like ability to direct traffic across multiple hops as directed by the resource-based routing algorithm.

A method to implement MPLS-TE is described on Cisco's site [20] as follows:

> One approach to engineer a backbone is to define a mesh of tunnels from every ingress device to every egress device. The IGP, operating at an ingress device, determines which traffic should go to which egress device, and steers that traffic into the tunnel from ingress to egress. The MPLS traffic engineering path calculation and signaling modules determine the path taken by the LSP tunnel, subject to resource availability and the dynamic state of the network. For each tunnel, counts of packets and bytes sent are kept. Sometimes, a flow is so large that it cannot fit over a single link, so it cannot be carried by a single tunnel. In this case multiple tunnels between a given ingress and egress can be configured, and the flow is load shared among them. [20]

D. PREVIOUS STUDIES OF BGP BLACKHOLE ROUTING (BGP BHR)

There have been a few studies carried out that talk about the analysis of DDoS mitigation with BGP BHR. The most complete and analytical is N. Stamatelatos' thesis, *A Measurement Study of BGP Blackhole Routing Performance*.[2] There is also a second study, V. Puri's *Automated Alerting for Blackhole Routing*,[3] which extends the research done by N. Stamatelatos' thesis.

Stamatelatos used a real test-bed network to evaluate the effectiveness of various methods of BHR. The performance metric chosen by Stamatelatos was router response time, router CPU load, and link load. He stress-tested three implementations of the BHR concept in a lab environment.

In Stamatelatos' study, a given DDoS attack had been positively identified by either an automated system or a human operator. The author recognized, in the "Future Work" section, [2] that the ability to automatically identify an attack using an IDS/IPS system would greatly improve the performance of BGP BHR and suggested the research in this field as an area for future work.

This suggested work is what Puri's thesis centered on. Puri managed not only to select and configure an appropriate IDS to detect a distributed denial of service (DDoS) attack; but to also integrate this detection capability into an enhanced BHR system, by having the IDS directly cue the "trigger router" that sends the null—blackhole—route update to all border routers. The result is a working implementation of a fully automated attack-detect-react-protect BHR system.

1. How BGP BHR Works

A BGP BHR system is one mechanism used to mitigate DDoS attacks. It uses a feature of almost all existing routers, the Null0 interface, in combination with the BGP routing protocol in order to drop undesired packets destined to a specific host.

The Null0 is a pseudo-interface that every router has by default. It is always up but can never actually forward or receive traffic. Whenever a packet is routed to Null0, it will be dropped. The purpose of the interface is to discard unwanted traffic.

The configuration for applying BHR is relatively simple. The basic requirement is a static route of the destination IP address to be discarded. This configuration for Cisco routers is shown in Stamatelatos' thesis as:

```
interface  Null0
no  icmp  unreachables
ip  route  127.0.0.0  255.0.0.0  null  0
```

Traffic is sent to the Null0 interface, and since there is no real host to receive the packets, ICMP Unreachable replies are submitted by default. To

prevent this unnecessary traffic, the first two lines from the previous configuration's example are used. The lines first specify the interface and then configure the router to not create ICMP Unreachable replies for this interface. The third line is the static route. In the above example, the packets that have as their destination the subnet 127.0.0.0/8 will be forwarded to the Null0 interface.

The Border Gateway Protocol (BGP) is the most popular routing protocol used between Autonomous Systems (AS). It is very powerful and gives network administrators many options in applying routing policies. When used inside an AS, it is called an internal BGP (iBGP). Routers that speak BGP establish a TCP connection between themselves, so that the exchange of information is reliable.

In BGP BHR, blocking malicious traffic is tried as early as possible. The most proper place to block malicious traffic is at the border routers where the traffic enters the network. By discarding traffic at that point, the network is protected, since no undesired traffic travels inside the AS. The basic implementation of BGP BHR requires a pre-configuration of all border routers with a static route entry to the Null0 interface. A router inside the AS is also configured to work as a trigger; it communicates with the border routers using iBGP.

To apply BHR, a special static route to the IP address of the victim needs to be added to the routing table of the trigger router. The static route contains more information under a "tag." Among this information, the most important is the "next-hop," which for BHR needs to be an IP address from the private subnet IP addresses already configured at the border routers. The trigger will automatically advertise the static route to the border routers, using an iBGP route update advertisement, and the border routers will update their routing table with the new entry, forcing all traffic destined to the victim to be routed to their null interface. To stop BHR, the static route at the trigger router is removed and the router will send out a route withdrawal to all border routers, again via iBGP.

The BGP BHR is not a perfect defense against DDoS attacks. Its most significant limitation is that it blocks traffic based only on an IP address. It cannot be more discrete in its filtering, for example, by dropping only telnet or HTTP packets going to the victim. Another drawback is that it is very hard to bypass or provide exceptions to the filtering, since to do so the router's forwarding table has to be bypassed.

There are many variations to the basic Blackhole routing technique, all of which can be categorized as one of two basic implementations: the Remote-Triggered (RTBH) and the Customer-Triggered. The main distinction between the two is the origination of the filtering command. The RTBH routing can be further divided into either destination-based routing or source-based routing, depending on what information (the source or the destination IP address) is used to block traffic.

The following sub-sections briefly discuss the network setup followed for both previous researches.

2. Lab Setup/Test Bed

a. Lab Setup in Stamatelatos' Research

Stamatelatos evaluated the performance of BHR methods in the lab with three real-time test-bed networks. He selected seven routers to simulate the various environments that depict the real-time AS. Stamatelatos utilized his chosen performance metrics in his test beds. A brief discussion of the three test-bed networks he used is as follows:

(1) Test-Bed Network #1. The main task of this test-bed network was to evaluate the performance of both the methods of remote-triggered BHR, i.e., destination-based and source-based. Stamatelatos simulated an AS environment with three border routers, two internal routers, and one trigger router. In this test bed, malicious traffic would approach the AS from different sources. In addition, traffic would also traverse through different border routers.

Once the attack began, the trigger router inside the AS was configured to advertise either source-based or destination-based BHR to evaluate both the techniques.

(2) Test-Bed Network #2. The main purpose of test-bed network #2 was to evaluate customer-triggered BHR and then compare its performance with remote-triggered BHR. Stamatelatos simulated this test bed by maintaining the same topology as discussed in test-bed network #1. The only difference was the positioning of the trigger router. The trigger router was placed in line with the target-host to simulate the customer network.

(3) Test-Bed Network #3. The purpose of test-bed network #3 was to evaluate the performance of BGP BHR in a network where the routers have sufficient CPU capacity but some of the internal links of the victim's network become congested during an attack. The researcher simulated this to evaluate performance when the limiting factor could be the link load, not the router CPU load. He utilized five routers, one of which is Juniper router with a relatively high CPU capacity, to simulate this test bed. The topology was different from test-bed networks #1 and #2.

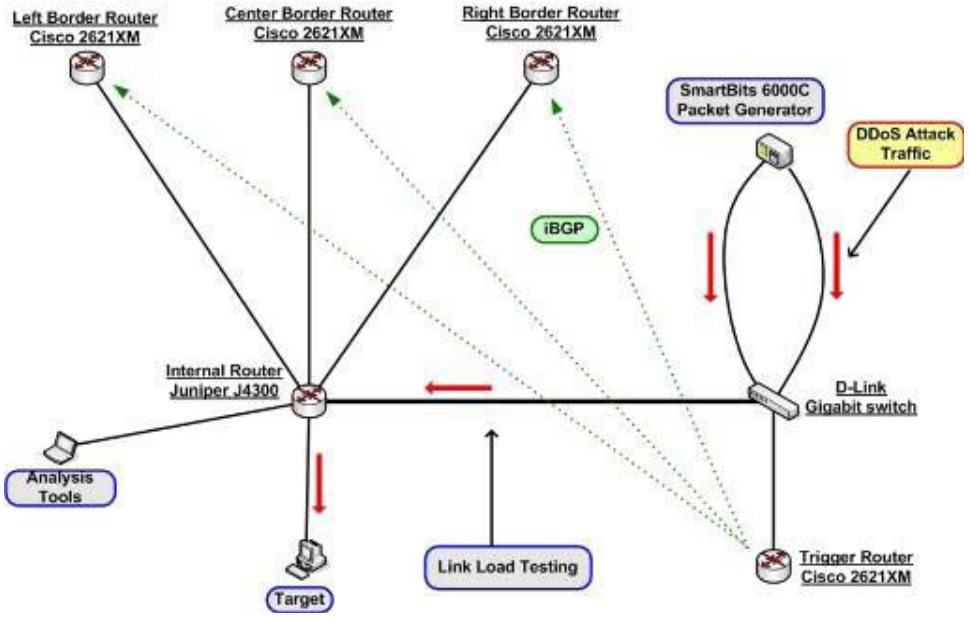


Figure 8. Stamatelatos' topology for test-bed #3 (From: [2])

b. Lab Setup in Puri's Research

Since Puri's work continued Stamatelatos' study, he had to select one of the above test-beds and techniques. He finally chose to work with the customer-triggered BHR technique.

He used a test-bed close to Stamatelatos' test-Bed Network #3, but simplified. More specifically he used two instead of three border routers for his AS. Instead of that, the rest of the components were placed by using almost identical topology.

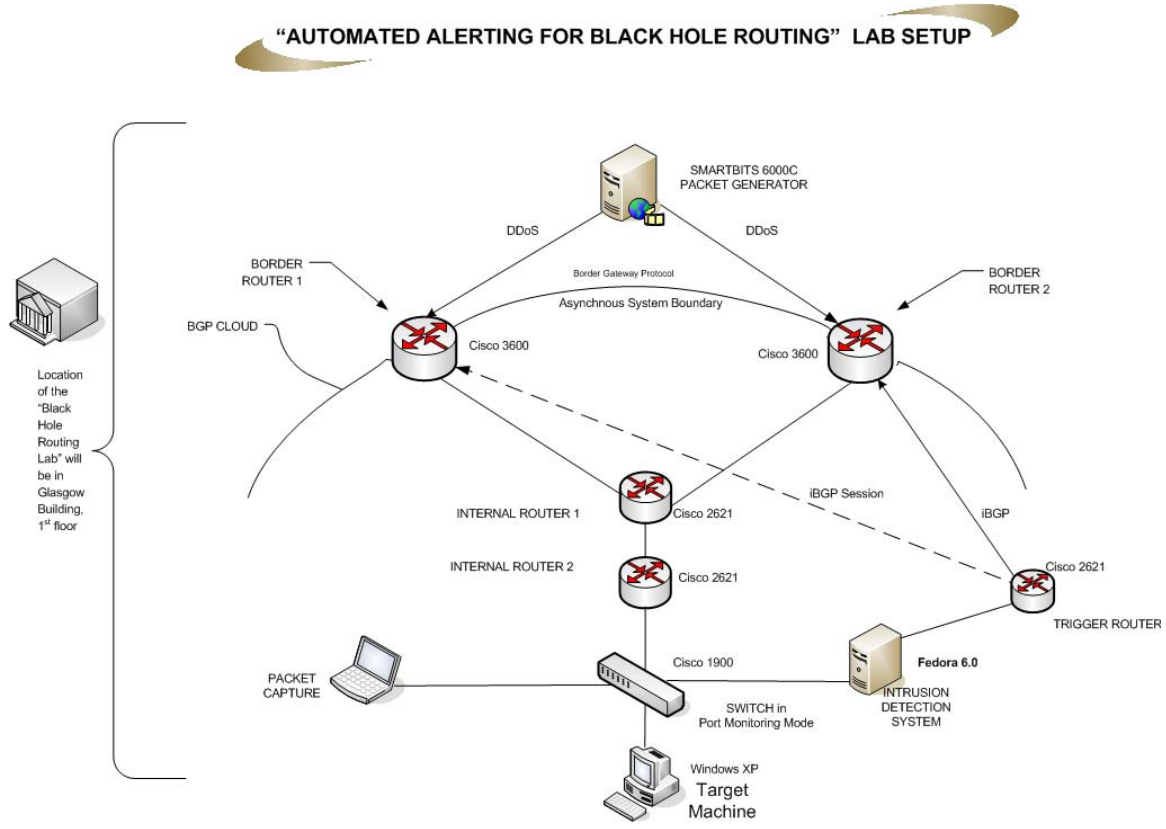


Figure 9. Puri's test-bed (From: [3])

3. Research Conclusions

a. Stamatelatos' Research Conclusions

- Resource overload may disrupt the BGP session between the trigger router and a border router and thus degrade the performance of BGP BHR.
- Customer-triggered BHR is not as effective as other techniques.
- Destination-based BHR performed best in test-bed simulations.
- The BHR would be totally inefficient if applied 40 seconds or more after the DDoS attack initialization (especially with high link load).

b. Puri's Research Conclusions

- The BHR proved to be one of the fastest ways to mitigate DDoS attacks on the network. Once an attack was detected, the system mitigated the DDoS attack in close to 20 seconds.
- The automation of BHR is not only an adaptable and useful technique, but it is also an efficacious and productive technique to mitigate DDoS attacks.
- Though BHR cannot be the sole solution to mitigate a D/DoS attack, it is recommended that the BHR solution be one of the mechanisms available to safeguard the target(s) and network resources from annoying D/DoS traffic within an AS.

4. Comments on Prior BGP BHR Work

Stamatelatos' thesis was focused on the BGP BHR performance by assuming that a DDoS attack had previously been recognized. Hence, the first shortcoming of his study was the absence of IDS.

Another shortcoming was the absence of automation. He manually added the static route to the trigger router to advertise the null route. When a network is under attack, time is critical. By manually typing a command of 35 characters extra delay time to the system's response is added, no matter how quickly one can type. If the time needed to connect with the server via telnet is also added, this approach proves to be unrealistic for real world systems.

Another major shortcoming in that research was in its conclusion. Stamatelatos concluded that customer-triggered BHR is least effective. He did not explore how this technique could be effective.

Puri's thesis actually extended Stamatelatos's work and addressed the issues that Stamatelatos' thesis had. At the beginning an IDS (Snort®) was employed in order overcome the first shortcoming as stated above in this paragraph. Second, he added automation (SnortSam) in order to add the static route to the trigger router to advertise the null route. He also used customer-triggered BHR and he proved that this technique is as effective as the remote triggered is.

Even if Puri had managed to overcome the major disadvantages that Stamatelatos' approach had, both of those studies still have not overcome the significant disadvantages that the BGP BHR technique has. The main disadvantages are:

- All the traffic flow, malicious or not, from each edge router is discarded during the attack. In other words, the attacker ultimately still wins. The victim server is no longer reachable from any other AS and so there is a Denial of Service.

- Which router to block or not cannot be determined, thus all routers have to be blocked, including those that are connected to a secure network.

- The discarded traffic is lost forever. It cannot be analyzed and perhaps "cleaned" in a dedicated place.

- A false positive response would result in a self-inflicted DoS attack.
- The response time of 20 seconds is acceptable within a relatively small network, but for a wide AS better performance has to be achieved.

These shortcomings have been overcome in this thesis research by engaging the MPLS-TE technique in combination with the services of IDS and the automated process of route advertisement that Puri used in his work.

E. EXISTING MPLS-TE TECHNIQUES FOR DDOS MITIGATION

Although it has great advantages, only two proposed techniques were found in the literature about the usage of MPLS-TE for DDoS mitigation.

1. MPLS-based Traffic Shunt

The first of them was presented during the 28th North American Networks Operators' Group's (NANOG) meeting in June 2003 in Salt Lake City, Utah [21]. The working team consisted of Yehuda Afek from Riverhead Networks, Roy Brooks from Cisco Systems and Nicolas Fischbach from COLT Telecom. The last participant presented the same work in September of the same year during the 46th Réseaux IP Européens Network Coordination Center's (RIPE NCC) meeting in Amsterdam, Netherlands [22].

The title of their work was "MPLS-based Traffic Shunt." With their presentation they proposed a new protecting method against DDoS by the usage of MPLS benefits in combination with the establishment of an "Inspection Device." The Inspection Device is actually a sinkhole router with a sinkhole server. A sinkhole router does exactly what a border router does when BGP BHR is used. The difference is that in this case and after the attack's detection a static route on a preselected core router (sinkhole router) is added which sends all the traffic destined for the victim to a dedicated interface as the Null0 in the BGP BHR method. This time, however, the BGP advertises that the victim is now connected on the sinkhole router. The sinkhole routing method adds an overload

to the network, since it carries all the malicious traffic through the network, but it provides the ability for a centralized inspection (on the sinkhole server) of the traffic—forensics.

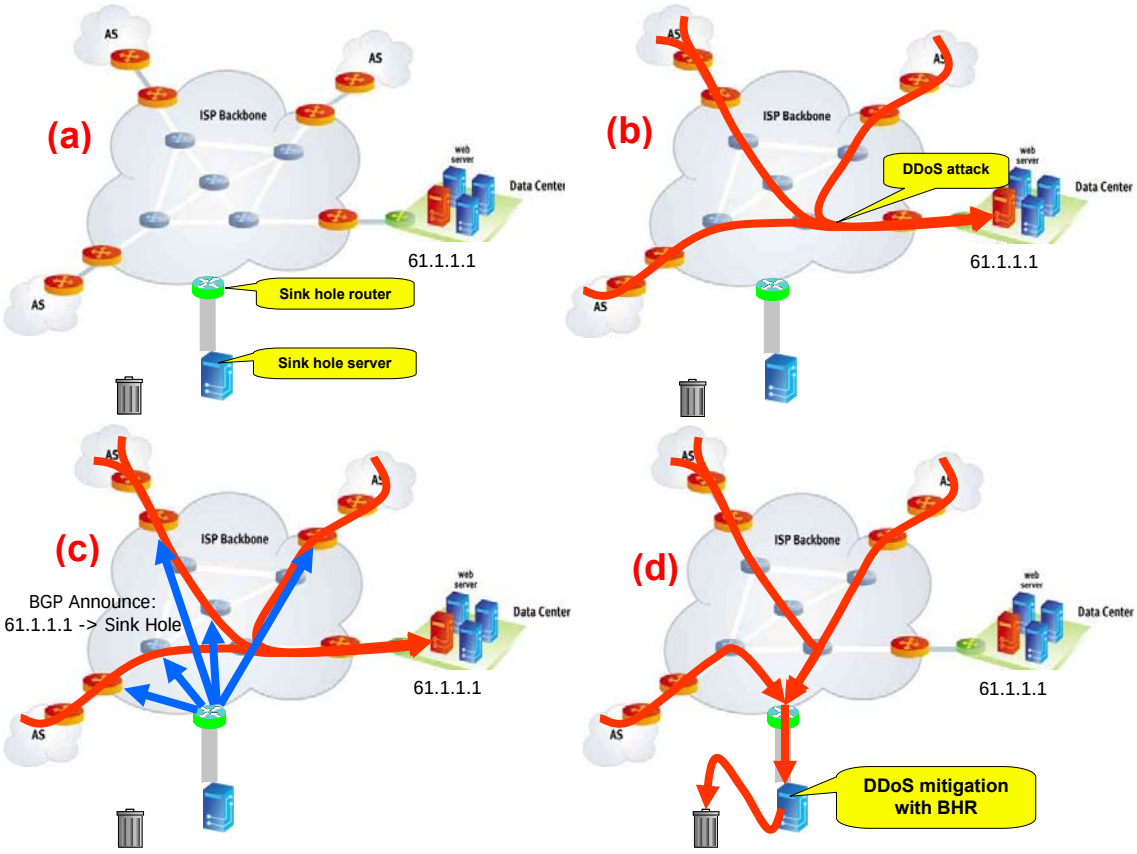


Figure 10. DDoS mitigation with the Sinkhole router technique (After: [21])

They proved that the sinkhole technique combined with traffic engineering techniques could provide a new capability. They could redirect the inspected and cleaned traffic back to the victim through the same network. With this new capability the network completely addresses the DDoS attack. Specifically, the MPLS-based traffic shunt has the following advantages [22] against the sinkhole method:

a. The MPLS-based method is bi-directional, which means legal traffic can be sent back to the target, losing only a small amount of non-malicious packets during the attack instead of all of them.

b. Since it used preconfigured tunnels, it does not add any overhead to the routers. The sinkhole routing without MPLS is based on IP techniques which add more routing complexity.

c. No additional software or hardware is required, since the routers employed already support MPLS.

In order to achieve their configuration, the team proposed the employment of tunnels from the peering/upstream routers to the inspection device and from the inspection device to the end system. They provided the following limitations [22] that this technique implies:

- Careful setup is required to avoid loops.
- Returned traffic must not pass through a peering router.
- Processing overhead for the sinkhole server is added.

They introduced two different methods to implement their MPLS-based traffic shunt. The first one was with pure MPLS using proxy LSP, which is going to be implemented in this thesis work, along with iBGP routing protocol in correspondence with the IP-based sinkhole technique described previously in this section. The main difference is that now the sinkhole server will be replaced by a “Cleaning Center” which has the capability to clean the traffic, drop the malicious packets and redirect the clean packets to their original destinations using the MPLS-TE attributes.

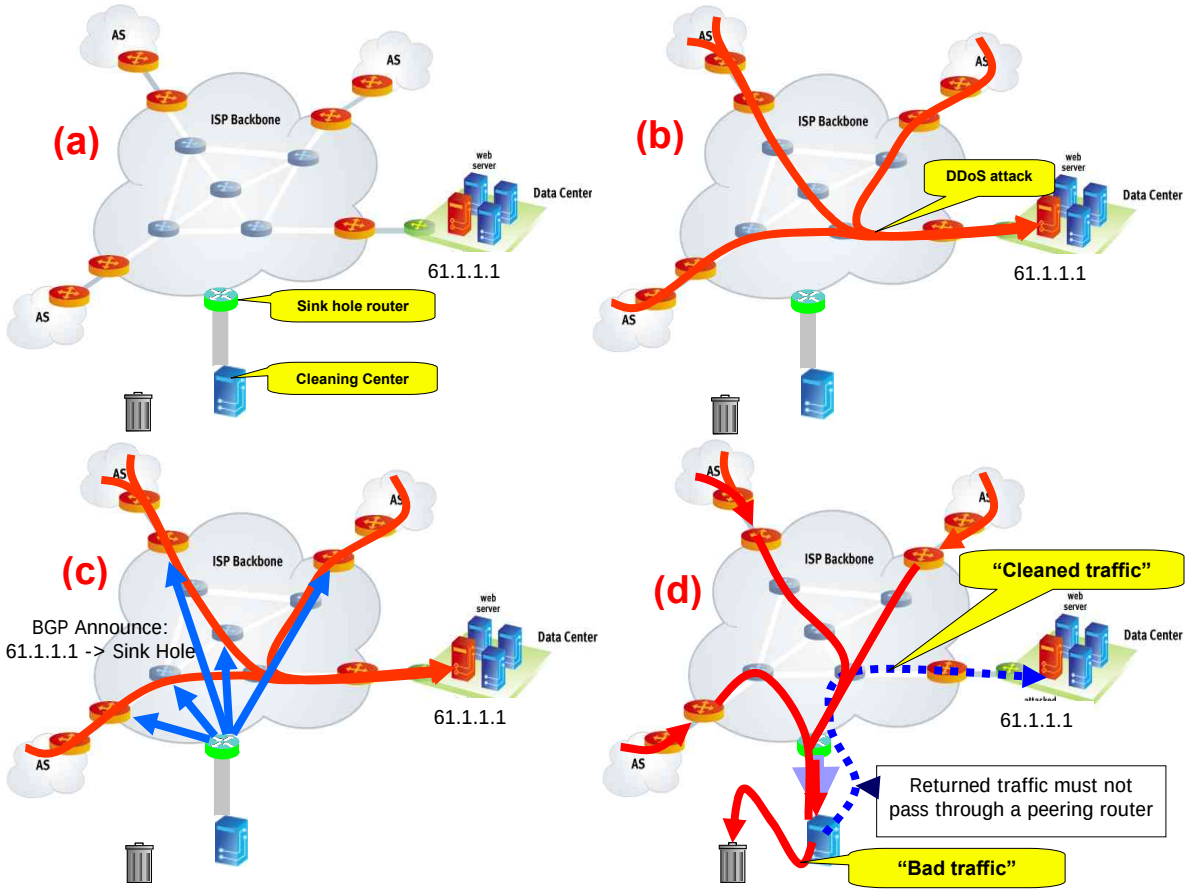


Figure 11. DDoS mitigation with the MPLS-based traffic shunt technique (After: [21])

The second proposed method from this first team was MPLS Virtual Private Networks (VPNs) using Virtual Routing and Forwarding (VRF), which actually is not a traffic engineering technique, and thus is out of the scope of the research for this thesis.

This author could not find any official report about the above methods, except for a power point file from their presentations at NANOG28 and RIPE46, even if after contact with the authors. So, there is no information as to how they achieved the attack detection, in other words, what IDS they used, or what kind of automation they used to trigger their sinkhole router.

At their presentations' conclusions [22] they stated that their techniques were:

- Actually deployed, not only in the lab.
- Proved easy to deploy, maintain and use.
- Improved DDoS detection, mitigation and analysis/post-mortem in conjunction with Netflowbased detection solution and customer profiling (filtering templates).

2. Sinkhole Routing with BGP Group Attributes

The second MPLS-TE technique found in the literature for DDoS mitigation was issued as a white paper in January 2007 by Huawei Technologies Co. Ltd. with the title “Technical White Paper for Sinkhole Routing”.[23]

The authors here used also the already shown sinkhole router method, but this time they went one step further. They focused on the usage of the group attributes of BGP routing protocol in order to achieve great scalability in their solution. They confirm in their paper that with their configuration, based on the BGP group attributes, they can achieve better performance of the protecting system.

This solution needs to assign a special group attribute value to all area border routers that may lead attacks in the ISP autonomous area in advance. With this technique, each border router is assigned a specific group attribute value. If a received route update report carries special group attributes assigned to this router or group attributes specifying all border routers, it will change the next hop attribute into the network segment address of a specific RFC 1918. So, traffic can be redirected only from attacked border routers and not from all of them. As a result, legal traffic on the routers not attacked will access the attacked host along a normal path, but the area border router of the attack entrance will block illegal traffic to reduce attack influence. At the same time since the route information of the core route remains unchanged, the access from the inside of the ISP to the objective host will not be affected.

With this solution they reduced the processing overhead of the sinkhole router and server (cleaning center). The proposed technique is actually an improved version of the above MPLS-based traffic shunt [21] technique.

The Huawei paper [23] does not make any reference to how to configure the MPLS-TE, the used security analysis/record facilities and the way the automated response can be achieved. Instead of that, they provide an example which gives the reader an abstract idea of how the techniques should be configured and should react as in the following figures.

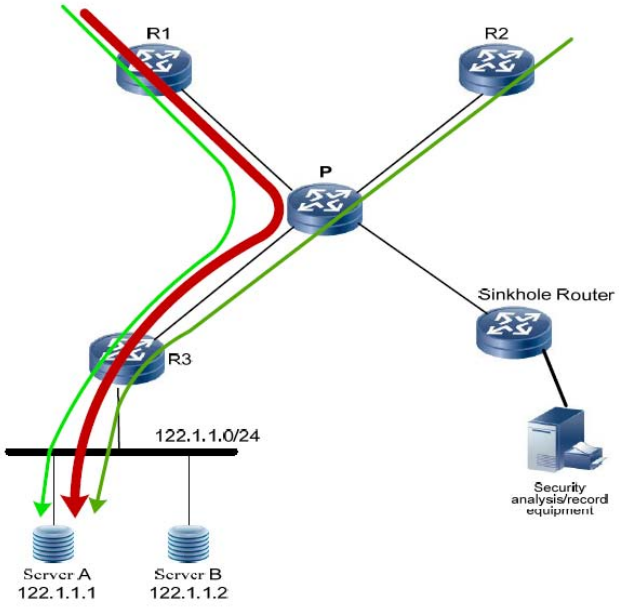


Figure 12. Attack to service A. (From: [23])

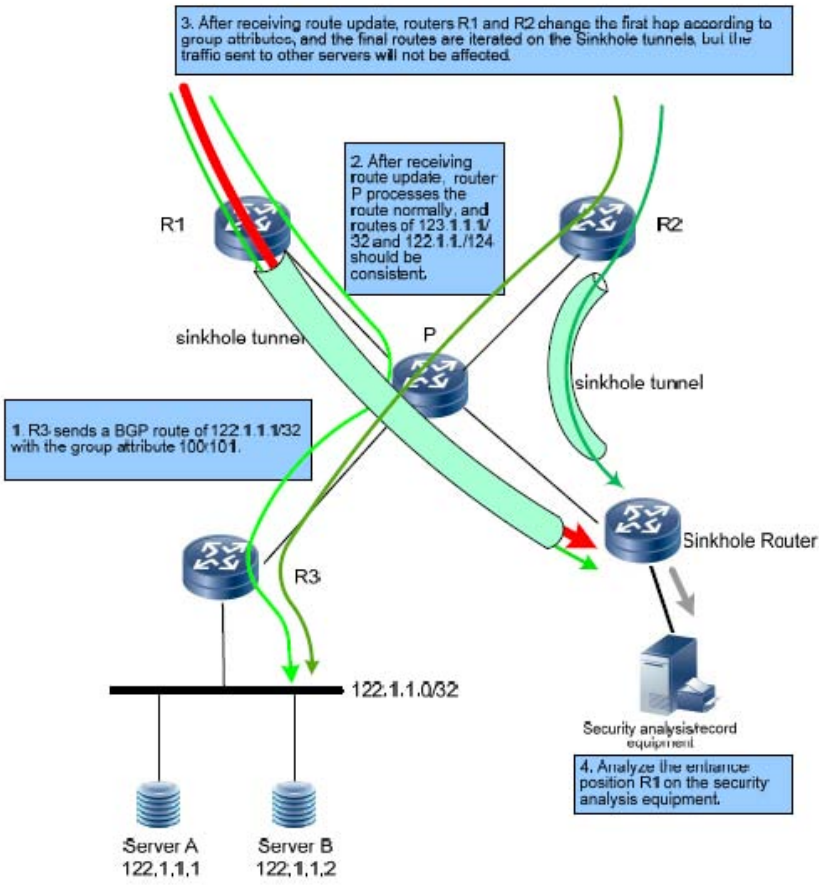


Figure 13. Lead all traffic to server A to the sinkhole router to make traffic of other servers normal. (From: [23])

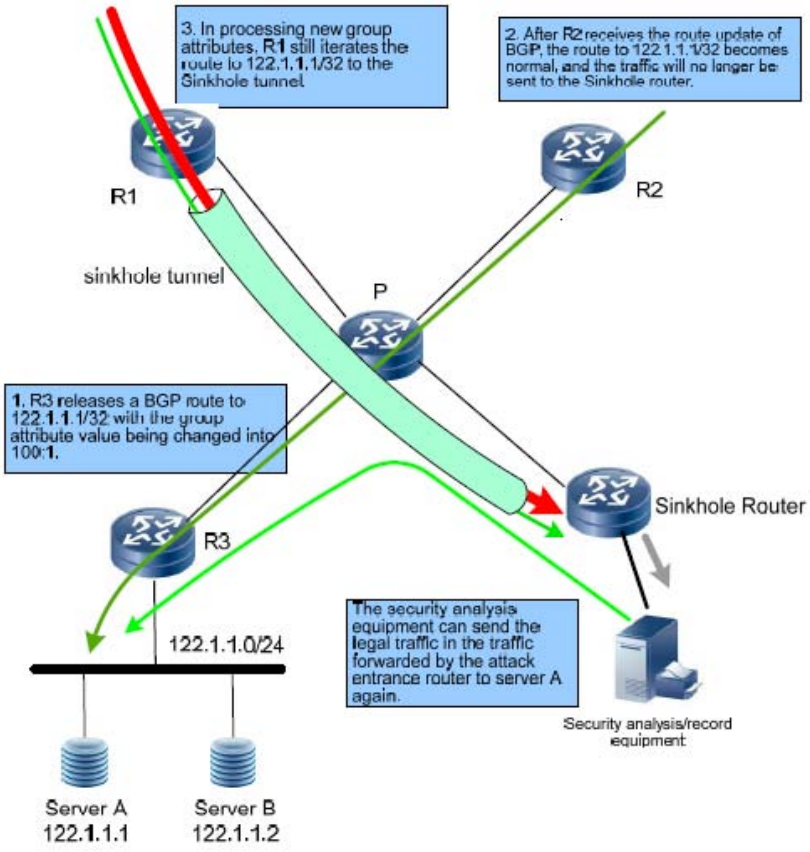


Figure 14. R3 releases a route again and the route of R2 at the non-attack entrance becomes normal.(From: [23])

In their summary part of the paper the authors conclude that the sinkhole routing technology combined with the enhanced BHR technology triggered by the BGP can reduce the DDoS's attack damage. They continued by stating that when this solution is integrated, in parallel with the ACL technology, the results are reduced network workload and better analyzed traffic.

3. Comments on Prior MPLS-TE Work

The above techniques are based on the same idea of sinkhole routing. The second idea can be seen as an improved version of the first. Both use BGP routing protocol to force the edge routers to redirect the traffic destined to the victim at the sinkhole router. After this point, the already preconfigured tunnels

route the traffic to the sinkhole router. This approach implies an extra overload to the routers and increases the response time. In both proposals, there is a lack of real world results and there is no reference to the techniques' performance. In the first approach a reference to an actual deployment is done, but no other details are provided. Both of the works claim that better results were achieved, but they do not provide any evidence, or any comparison with the previous techniques.

The technique used in this current study borrowed the sinkhole routing idea in combination with the MPLS-TE, as in the previous two techniques. The main difference is that this research did not use BGP protocol. The installed routing protocol was OSPF and the redirection of the traffic to the sinkhole router made through telnet sessions between the trigger server and each of the edge routers.

THIS PAGE INTENTIONALLY LEFT BLANK

III. SETUP OF TEST-BED

A. CHAPTER OVERVIEW

This chapter describes the laboratory set-up of the MPLS-TE network used for this thesis research. Section B presents the overall network architecture of the MPLS-TE laboratory setup and the basic configuration for the LER and LSR routers. This is followed by a detailed description of the various parameters of interest and the required hardware and software configuration for evaluating the effectiveness of an MPLS-TE based solution against DDoS attacks. The last Section describes the software tools used to detect a DDoS attack and trigger an automated response against the attack.

B. NETWORK'S CONFIGURATION

1. General

For a fair comparison with previous BGP BHR studies, the author chose to build a test-bed close to that used by Puri. The main difference in this current test-bed is the packet routing technique which, in this case, is MPLS-TE over IP instead of plain IP. For the same reason, the author elected to use identical or newer versions of the software in Puri's attack detection and response system.

2. Hard ware

The following devices are used for this research test bed network:

- Four Cisco routers with IOS C3620 software, Versions 12.2(17a), 12.2(24a), 12.2(29) and 12.2(3) with four 10 Mbps Ethernet Interfaces, used as Border Routers.

- One Cisco router with IOS 3600 software, Version 12.2(3) with four 10 Mbps Ethernet Interfaces and one 100 Mbps Fast Ethernet Interface used as an internal router.
- One Smart Bits 6000C Performance Analysis System of Spirent, for packets' generation.
- Three desktop PCs with Windows XP SP 2. One is used for Smart Bits' and routers' configuration. The second one as an attack monitor connected with LER2. The third one is used as a target machine.
- One desktop PC with Fedora 8.0 is loaded on with the IDS.
- One LAN-3321A TeraMetrics XD module with two 10/100/1000 Mbps Ethernet Copper ports and two 1 Gigabit Ethernet Fiber ports installed on the Smart Bits 6000C system. Both the copper ports are used to simulate a D/DoS attack.
- One Hub used to create a subnetwork between the IDS's and the target's machines.

3. Sofw are

The applications used for this research are as follows:

Smart Window version 7.70.128, for use with the Smart Bits 6000C system to generate attack traffic.

CommView version 6.0 of Tamosoft, for crafting custom ICMP packets.

Wireshark version 1.0.0 and 1.0.3 Network Protocol Analyzer on Windows XP machines.

Wireshark version 1.0.3 Network Protocol Analyzer on Linux 2.6.26.5-28.fc8 for monitoring the target network traffic.

Snort® version 2.6.1.3 as the Intrusion Detection System (IDS).

SnortSam added as a plugin program to the Snort® package in order to achieve automated detection of and response to the attack.

4. Topology and MPLS Tunnels

The following figure shows the implementation of the MPLS-TE network set-up in the laboratory. The MPLS-TE network is formed by five routers. One LSR router performs label switching and emulates the core of an MPLS network backbone. Four LER routers are entry and exit points to the network. Each LER router is directly connected to the LSR router. The LER2 and LER3 routers are connected, also, to the Smart Bits 6000C through its LAN-3321A TeraMetrics XD Ethernet Copper ports. The DDoS attacks are launched from those two points. The LER4 is connected with the target's sub-network. The target's sub-network includes one Windows XP desktop, acting as the target host, and another Fedora Linux machine, acting as the IDS/automation system. Both of those machines are connected through a HUB to the LER4. Finally, LER1 is connected with a Windows XP desktop, which simulates the cleaning center.

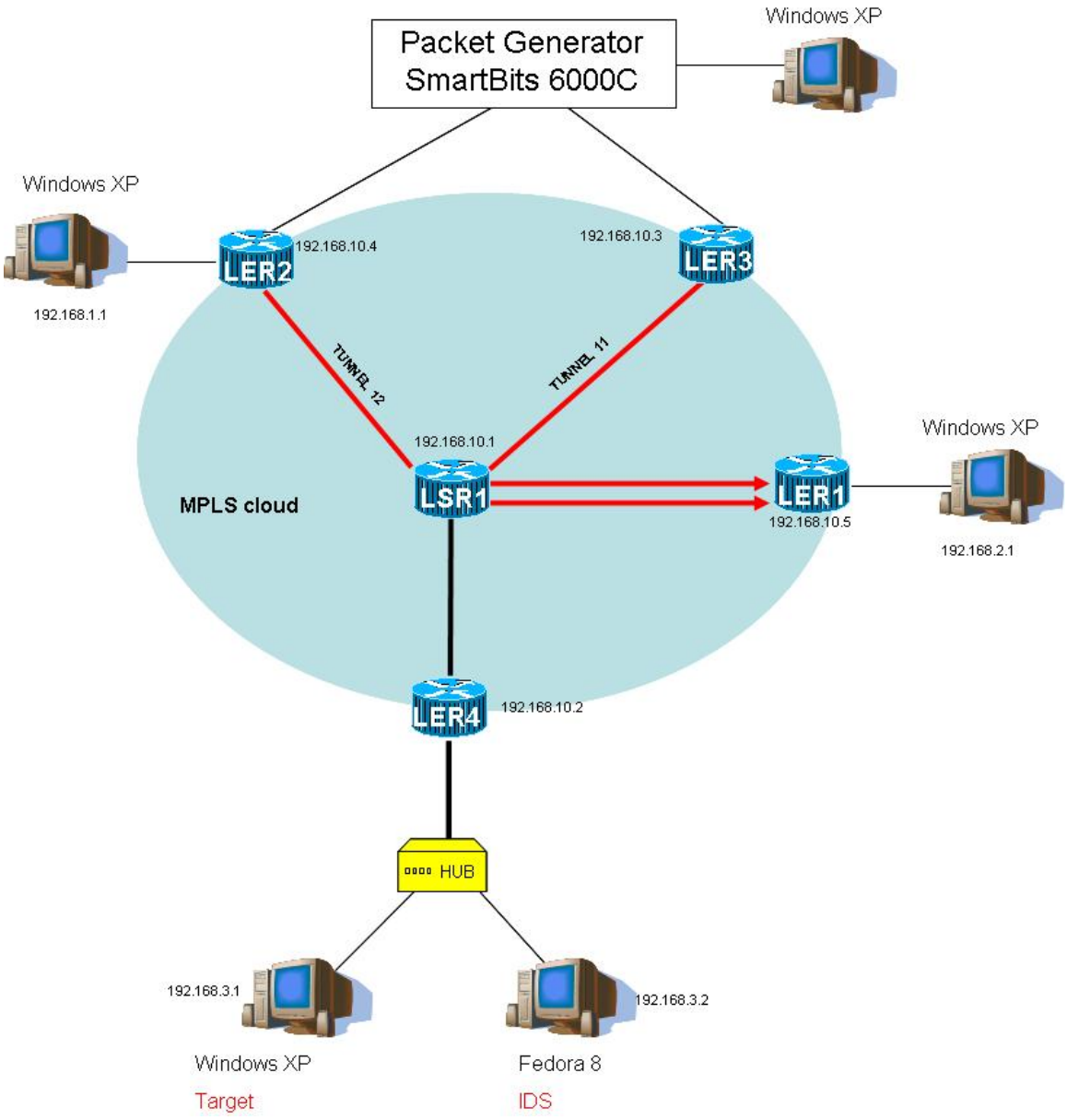


Figure 15. Network topology

Nine MPLS tunnels have been preconfigured with MPLS-TE parameters and their basic characteristics are described in the following table:

Tunnel From		To	Bandwidth
10	LER3	LER4	4800 kbps
11	LER3	LER1	4800 kbps
12	LER2	LER1	4800 kbps
13	LER2	LER4	4800 kbps
14	LER4	LER1	100 kbps
15	LER4	LER2	4800 kbps
16	LER4	LER3	4800 kbps
17	LER1	LER2	4800 kbps
18	LER1	LER3	4800 kbps
19	LER1	LER4	100 kbps

Table 1. Preconfigured MPLS-TE tunnels

The purpose of tunnels 11 and 12 is to divert the attack packets from their ingress routers to the cleaning center. The rest of the tunnels are used for normal traffic.

When an attack is detected by the IDS installed on the Fedora machine, the plug-in program to the IDS is activated and starts a telnet session at the beginning with the cleaning center's LER router – LER1 – and adds a static route as follows:

```
ip route 192.168.3.1 255.255.255.255 interface ethernet1/0
```

Sequentially, it starts telnet sessions with LER2 and LER3 – one at a time – and adds the following static routes:

`ip route 192.168.3.1 255.255.255.255 interface Tunnel12` on LER2 and, `ip route 192.168.3.1 255.255.255.255 interface Tunnel11` on LER3. The ip address 192.168.3.1, was added earlier as a

second ip address, on the ”clean center” machine connected on interface Ethernet 1/0 of LER1. Consequently, LER2 and LER3 will forward all traffic destined to the target host for the cleaning center. From this point forward the DDoS attack can no longer impact the target host, similar to what happens when using BGP BHR methods. However, this technique provides one significant advantage. The traffic from the edge routers is not simply discarded. It is directed to the cleaning center, where it can be analyzed and cleaned and the legal part of it can be redirected back to the attacked machine through the same MPLS-TE network used during the attack.

5. Router Configuration (Edge, Core)

This Section shows the configurations required to set up the LER and LSR routers for the MPLS-TE test-bed. The configuration files for the rest of routers in the test-bed are presented in Appendix A.

a. Installing LER Router

The four LER routers are Cisco 3620 routers running the Cisco Internetworking Operating System (IOS) version 12.2. The connection between the LER routers and the LSR router are established using Ethernet interfaces. Table 2 shows the MPLS-TE configuration for one of the LER routers – LER2.

Configuration of LER router – LER2
Current configuration : 1872 bytes ! version 12.2 service timestamps debug uptime service timestamps log uptime no service password-encryption ! hostname LER2 ! ! ip subnet-zero ! !

```
!
ip cef
mpls traffic-eng tunnels
call rsvp-sync
!
!
!
!
!
!
!
!
interface Loopback0
 ip address 192.168.10.4 255.255.255.255
!
interface Tunnel12
 ip unnumbered Loopback0
 tunnel destination 192.168.10.5
 tunnel mode mpls traffic-eng
 tunnel mpls traffic-eng autoroute announce
 tunnel mpls traffic-eng priority 7 7
 tunnel mpls traffic-eng bandwidth 4800
 tunnel mpls traffic-eng path-option 1 explicit name sec-LSR1
!
interface Tunnel13
 ip unnumbered Loopback0
 tunnel destination 192.168.10.2
 tunnel mode mpls traffic-eng
 tunnel mpls traffic-eng autoroute announce
 tunnel mpls traffic-eng priority 7 7
 tunnel mpls traffic-eng bandwidth 4800
 tunnel mpls traffic-eng path-option 1 explicit name def-LER4
!
interface Ethernet0/0
 ip address 192.168.1.128 255.255.255.0
 half-duplex
!
interface Ethernet0/1
 description Connection to LSR1
 ip address 192.168.4.128 255.255.255.0
 half-duplex
 mpls traffic-eng tunnels
 tag-switching ip
 priority-group 1
 ip rsvp bandwidth 10000 10000
!
interface Ethernet1/0
 no ip address
 shutdown
 half-duplex
!
interface Ethernet1/1
 ip address 192.168.0.102 255.255.255.252
 full-duplex
!
```

```
router ospf 99
 router-id 192.168.10.4
 log-adjacency-changes
 network 192.168.0.0 0.0.255.255 area 0
 mpls traffic-eng router-id Loopback0
 mpls traffic-eng area 0
!
ip classless
no ip http server
!
ip explicit-path name sec-LSR1 enable
 next-address 192.168.4.1
 next-address 192.168.7.128
!
ip explicit-path name def-LER4 enable
 next-address 192.168.4.1
 next-address 192.168.6.128
!
priority-list 1 protocol ip high tcp telnet
priority-list 1 protocol ip low
!
!
dial-peer cor custom
!
!
!
!
line con 0
 password vordos
 login
line aux 0
line vty 0 4
 password vordos
 login
!
end
```

Table 2. MPLS-TE Configuration of LER Router – LER2

The "`tag-switching ip`" command in the router configuration enables MPLS for a network interface. It is an alternative to the "`mpls ip`" command available in newer Cisco IOS versions. In the sample configuration above, the "`tag-switching ip`" command is used for the network interface connecting to the LSR router.

As stated in the Background chapter each MPLS-TE router should use at least one layer three routing protocol of the Interior Gateway Protocol (IGP) type. The most commonly adopted IGPs for MPLS-TE are OSPF and IS-IS (link state

protocols) in MPLS configurations as they are the only two IGPs that support MPLS traffic engineering. The IS-IS uses new Type-Length-Values (TLVs); OSPF uses type 10 Link-State Advertisements (also called Opaque LSAs).[24] There is no strong reason to use one IGP over the other for the laboratory set-up. The OSPF is the author's selection for the configuration presented above. The next step is to enable the routing protocol (OSPF) to operate in the MPLS-TE environment by entering the commands "`mpls traffic-eng router-id Loopback0`" and "`mpls traffic-eng area 0.`"

In order to enable the MPLS-TE features of this test-bed, the command "`mpls traffic-eng tunnel`" shown in Table 2 is used. A tunnel's configuration starts with the command "`interface TunnelX,`" where X is the tunnel's number. Subsequently, the tunnel's destination must be specified with the command "`tunnel destination XXX.XXX.XXX.XXX,`" where the ip address is the destination LER's Loopback0 address. In order to enable the ReSerVation Protocol (RSVP) the command "`ip rsvp bandwidth 10000`" is entered on each concerned interface for non-zero bandwidth tunnels.

Then, the tunnels to be used for TE are set up. There are many options that can be configured for an MPLS TE tunnel, but the command "`tunnel mode mpls traffic-eng`" is mandatory. The "`tunnel mpls traffic-eng autoroute announce`" command met on this configuration announces the presence of the tunnel by the routing protocol. The priority of the tunnels has been set to 7, which is the highest possible value and corresponds to the lowest forwarding priority. The bandwidth is mostly specified to 4800 kbps with the command "`tunnel mpls traffic-eng bandwidth 4800.`" Only two tunnels have a different bandwidth of 100 kbps. They are used to connect the target's LER – LER4 and the cleaning center's LER – LER1, and hence, a smaller bandwidth is sufficient.

We define the name of the explicit route i.e., “def-LER4” with the command “tunnel mpls traffic-eng path-option 1 explicit name def-LER4.”

As can be seen in the above configuration, each tunnel is considered as a router’s interface. The “ip unnumbered Loopback0” configuration command allows enabling IP processing on a serial interface without assigning it an explicit IP address. That interface can "borrow" the IP address of another interface already configured on the router (the Loopback0 interface in this case), which conserves network and address space [25]. After each tunnel’s initial configuration the explicit route is defined in a hop – by – hop manner with the command “ip explicit-path name def-LER enable.”

In the test-bed, all nine tunnels are implemented by using the “explicit paths” method (i.e., manually by the administrator). The implementation of dynamic tunnels (automatically set up by the ingress LER), has been avoided since the diversion path should be clearly defined by the network’s administrator in order to lead the malicious traffic to the cleaning center, through a “safe” route.

The command “priority-list 1 protocol ip high tcp telnet” gives the highest priority to Telnet packets, while the command “priority-list 1 protocol ip low” gives a lower priority to the rest of the tcp packets. The command “priority-group 1” under the definition of interface “Ethernet0/1” dictates the interfaces to follow this priority arrangement.

Finally, the passwords to protect the router from unauthorized access are set up. The command “line con 0” sets the password to restrict configuration change with the command “enable” in a console window. The command “line vty 0 4” sets the password to control inbound Telnet connections. Both passwords, for simplicity, are set to “vordos” on all routers.

The configuration of the rest of LER routers is similar to the one in Table 2 except for the values of some parameters such as the IP addresses for the loopback interfaces, and IP addresses for network interfaces.

b. Installing LSR Router

The configuration of the LSR router is simpler than the LER routers because MPLS tunnels are already configured at the ingress edge routers. Like the LER routers, a Cisco 3620 router with Cisco IOS 12.2(3) is used for the LSR router. Table 3 shows the configuration for the LSR router—LSR1.

Configuration of LSR router – LSR1

```
Building configuration...

Current configuration : 1577 bytes
!
version 12.2
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname LSR1
!
!
ip subnet-zero
!
!
!
ip cef
mpls traffic-eng tunnels
call rsvp-sync
!
!
!
!
!
!
!
!
!
interface Loopback0
 ip address 192.168.10.1 255.255.255.255
!
interface Ethernet0/0
 description connection to Router LER1
 ip address 192.168.7.1 255.255.255.0
```

```
 half-duplex
 mpls traffic-eng tunnels
 tag-switching ip
 priority-group 1
 ip rsvp bandwidth 10000 10000
!
interface Ethernet0/1
 description connection to LER3
 ip address 192.168.5.1 255.255.255.0
 half-duplex
 mpls traffic-eng tunnels
 tag-switching ip
 priority-group 1
 ip rsvp bandwidth 10000 10000
!
interface Ethernet0/2
 description connection to LER2
 ip address 192.168.4.1 255.255.255.0
 half-duplex
 mpls traffic-eng tunnels
 tag-switching ip
 priority-group 1
 ip rsvp bandwidth 10000 10000
!
interface Ethernet0/3
 description connection to LER4
 ip address 192.168.6.1 255.255.255.0
 half-duplex
 mpls traffic-eng tunnels
 tag-switching ip
 priority-group 1
 ip rsvp bandwidth 10000 10000
!
interface FastEthernet1/0
 no ip address
 shutdown
 duplex auto
 speed auto
!
router ospf 99
 router-id 192.168.10.1
 log-adjacency-changes
 network 192.168.0.0 0.0.255.255 area 0
 mpls traffic-eng router-id Loopback0
 mpls traffic-eng area 0
!
ip classless
no ip http server
!
priority-list 1 protocol ip high tcp telnet
priority-list 1 protocol ip low
!
!
dial-peer cor custom
!
```

```
!
!
!
gatekeeper
 shutdown
!
!
line con 0
 password vordos
line aux 0
line vty 0 4
 password vordos
 login
!
end
```

Table 3. MPLS-TE Configuration of LSR Router – LSR1

6. Target

One Windows XP machine is selected as a target machine. The IP address 192.168.3.1 is assigned to this Windows machine. Wireshark is loaded onto this machine to capture the packets and to note the efficacy of the DDoS attack.

7. Traffic Generator

To test the effectiveness of the selected MPLS-TE technique, DDoS attacks for the test-bed network described above must be created. The hardware available for this task is the SmartBits 6000C Performance Analysis System of Spirent Communications, with one LAN-3321A TeraMetrics XD module with two 10/100/1000 Mbps Ethernet Copper ports and two Gigabit Ethernet Fiber ports. The system offers the ability to create customized layer-three and layer-four packets in IPv4 and IPv6 formats.

Furthermore, it provides the user with the capability to customize layer-two information (i.e., source and destination MAC address). All the ports of the module can operate in full or half duplex mode. The interfaces act as regular

hosts inside a network. To control the system, the SmartWindow version 7.70.128 Graphical User Interface (GUI) application is used. The figures below present the main screens of this application.

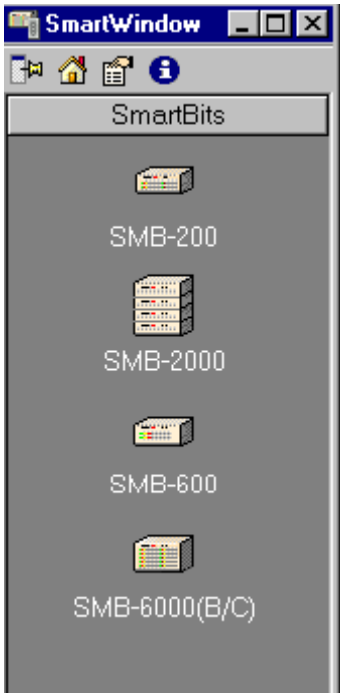


Figure 16. SmartWindow screen for device selection

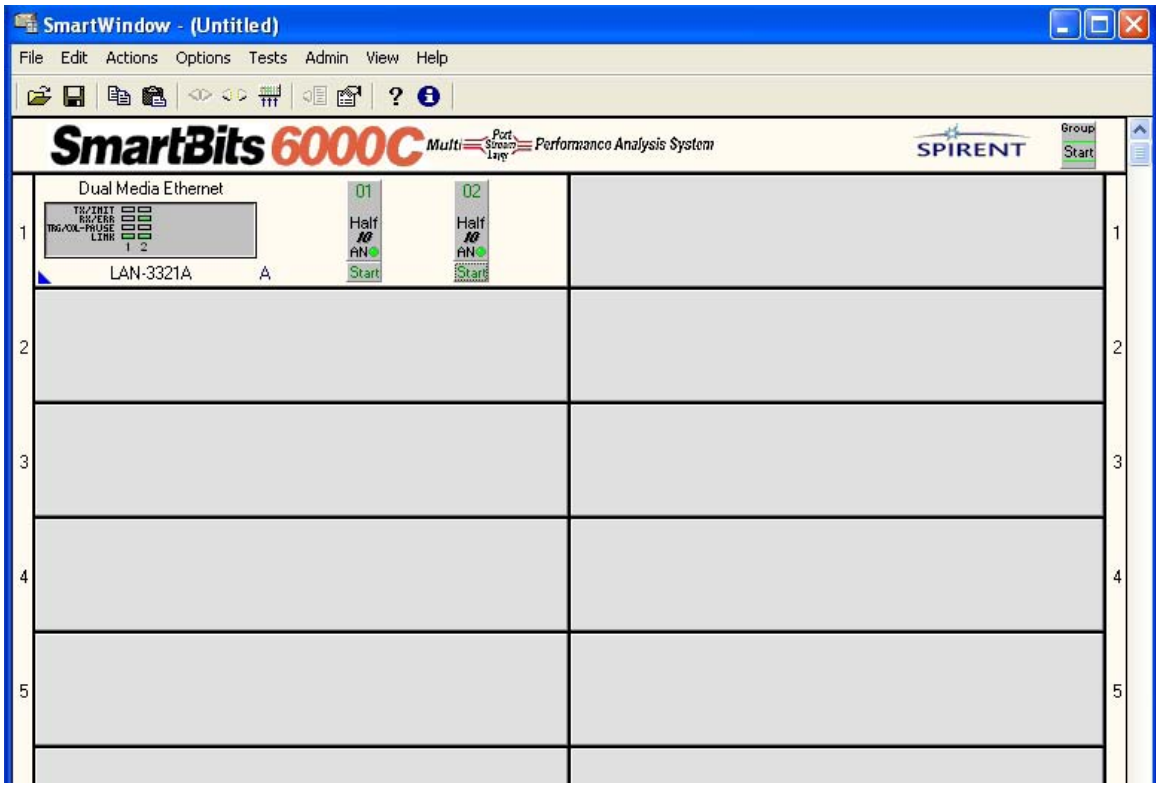


Figure 17. SmartWindow main screen for SmartBits 6000C device

Since the purpose of this study was to test the MPLS-TE network's reaction on a massive DDoS attack, a sophisticated attack does not have to be engineered. So, the author chose a simple attack to implement in the laboratory: an ICMP flood attack. As described in the Background chapter, in this kind of attack, the attacker sends a large number of ICMP_ECHO packets ("ping") to the victim system. An ICMP flood attack is very easy to be addressed by applying a simple rule on the router's ACL, which blocks all the incoming ICMP packets. However, since this attack was used in previous BGP BHR studies, this solution is followed for comparable results in this study.

To craft the ICMP packets used for the attack, CommView version 6.0 of Tamosoft and Wireshark are used. Once the desired ICMP packets are crafted, the Smart Bits 6000C system, with LAN-3321A TeraMetrics XD module with two

10/100/1000 Mbps Ethernet Copper ports, is used to simulate the DDoS attack. The in-depth explanation of this entire process is provided in Puri's thesis Appendix H [3].

C. AUTOMATIC INTRUSION DETECTION SYSTEM

1. IDS (SNORT®) Setup

This is one of the most important and critical components of the network test-bed. There are two basic types of IDSs on the market. The first type is the network based IDSs (NIDS) that is designed to monitor traffic for multiple hosts in the network. The other type of IDS used to detect changes or malicious activity for one specific host, is called a host-based IDS (HIDS). As Puri proved in his second chapter of thesis research [3], the most suitable type of IDS for this research is a NIDS. Since one purpose of this research is to compare the BGP BHR with the MPLS-TE techniques, the same IDS as in Puri's study – Snort® is selected. Snort® version 2.6.1 software is downloaded from the official Snort® site. This Snort® web site reference manual and Puri's directions were very helpful in setting up the alerter on this current network [26], [3]. Snort® i installed on Fedora 8.0. Snort® software versions are also available for Windows, Solaris, and others. Research has revealed that Snort® is most stable with a Linux-based environment. The following are the step-by-step details followed for setting up Snort.

a. Before Snort®'s Installation

Before Snort®'s installation, some basic network settings have to be performed, installed and configured the services needed to run. During this setup, the firewall is turned off for simplicity. While configuring the network, the following should be clearly configured:

(1) IP Address. One IP address is allocated to the alerter. The IP address provided to our Fedora machine is 192.168.3.2 and configured for

Ethernet which sniffs the traffic for network 192.168.3.0/24. Since this interface runs in promiscuous mode, there is no actual need for its IP address to belong to a specific network.

(2) Netmask. A network mask of 255.255.255.0 is used.

(3) Gateway. As default gateway is provided the LER router's interface address 192.168.3.128.

(4) DNS Server. In this scenario, a DNS server is not configured.

(5) Services. Ports 22, 23, 80, 443, and 3306 are enabled to support SSH, TELNET, HTTP, SSL and MySQL services in the Fedora box.

Before installing Snort®, the following required components are also preinstalled:

mysql, mysql-bench, mysql-server, mysql-devel, php-mysql, httpd, gcc, pcre-devel, php-gd, gd, mod_ssl, glib2-devel, gcc-c++, mysql-connector-odbc, mysql-server, libnet10-1.0.2a, libpcap-1.10

b. Installing MySQL and Snort®

At this stage, the Snort® is installed and the MySQL database to configure the Snort® alerts is configured. Furthermore, a few directories that would be used by Snort® are created.

Snort® version 2.6.1 is downloaded into "/" directory. The file name of the Snort® package is snort-2.6.1.3.tar.gz. This file is extracted and compiled as follows:

```
tar zxvf snort-2.6.1.3.tar.gz

cd snort-2.6.1.3

./configure - -with –mysql --enable-dynamicplugin

make all

make install
```

The above commands successfully installed Snort® in the Fedora machine. The third command indicates that Snort® is compiled with MySQL and enables dynamic plug-in to the program.

The up-to-date Snort®'s rules were found on its official website, which are downloaded into the /usr/local/src directory and are copied into newly created directories of Snort® as follows:

```
mkdir /etc/snort

mkdir /var/log/snort

mkdir /etc/snort/rules

tar zxvf /usr/localsrc/snortrules-snapshot-Current.tar.gz –C
/etc/snort

cp etc/*.conf* /etc/snort

cp etc/*.map /etc/snort

ln –s /usr/local/bin/snort /usr/sbin/snort
```

The following three commands created a Snort® user and user group in the snort directory.

```
groupadd snort

useradd -g snort snort

chown snort:snort /var/log/snort
```

In order to get Snort® up and running, a few configuration changes in a file called snort.conf, which exists within the /snort/snort-2.6.1.3 directory are required. This file is edited, the string "`var RULE_PATH`" is located and the variable is modified as follows:

```
var RULE_PATH /etc/snort/rules
```

Then, the following string “`database: log to variety of databases`” is located and the following line, directly after the commented lines, is added:

```
output database: log, mysql, user=snort password=password dbname=snort host=localhost
```

The line above tells Snort® to log the events in the MySQL database. Snort® is also provided with the details of the database. The database name is “snort,” the user name is also “snort,” and the password is “password.”

At this point, the database named “snort” in MySQL has been created. To achieve this, the following statements are issued:

```
mysql

SET PASSWORD FOR root@localhost=PASSWORD(‘password’);

create database snort;

grant CREATE, INSERT, SELECT, DELETE, UPDATE on snort.* to snort@localhost;

SET PASSWORD FOR snort@localhost=PASSWORD(‘password’);

exit
```

The Snort® package also contained the schema for various databases. These schema are stored in the snort-2.6.1.3 directory. The following commands activate the database schema:

```
/snort-2.6.1.3/schemas

mysql –p < create_mysql snort
```

So the database called snort has been created. Now, the Snort® installation can be tested by giving the following command:

```
/usr/local/bin/snort -c /etc/snort/snort.conf
```

The Snort® process creates the alert file under /var/log/snort/ on its own. The permissions of the alert file have to be changed so that the Snort® user can access that file. This is achieved by giving the following commands:

```
chown snort: snort /var/log/snort/alert
chmod 600 /var/log/snort/alert
```

c. Installing Snort®'s Graphic Interface

At this point, BASE and ADODB packages have to be installed. The ADODB package provides the interface between the GUI and the MySQL database. Additionally, the BASE package provides the graphical front end to the snort database. These packages are downloaded from sourceforge and are installed to ensure the proper functioning of Snort® and its customized Snort® rule:

```
cd /var/www/html
tar zxvf /root/adodb490.tgz
tar zxvf /root/base-1.2.7.tar.gz
chown apache base-1.2.7
service httpd restart
```

Now, the http service has been restarted and the BASE is configured by opening the browser with URL `http://localhost/base-1.2.7`.

The BASE setup program starts on its own. It prompts for the path to ADODB in the first step. The path name is given as `/var/www/html/adodb`. The next step is to enter the database name, database host, database user name, and database password. Exactly the same details as configured above in this section are entered. Then "`the submit query`" button on the screen is clicked. On-screen instructions in the setup script are followed to create the database tables used by the BASE application. When done, the "`Create BASE`

AG" button is clicked and the tables are created. The next screen is the login screen. The login credentials are entered and the BASE main screen appears as in the following figure:

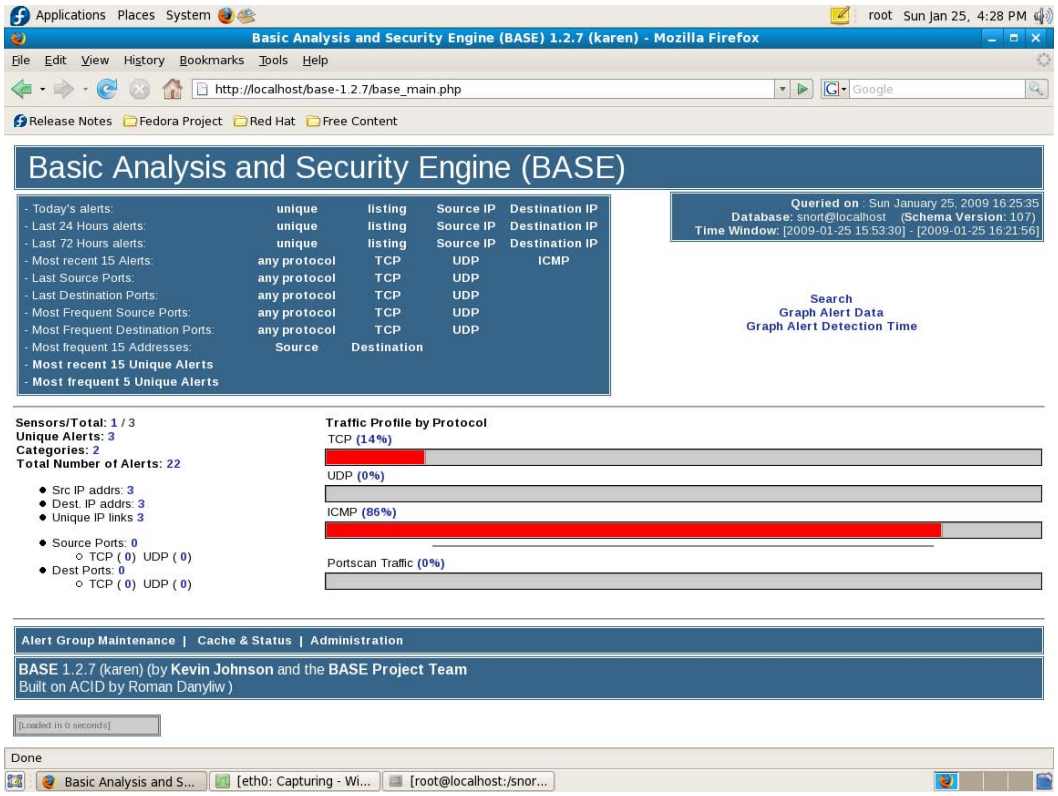


Figure 18. BASE snapshoot

To enable the BASE graphing capability the php-pear-1.6.2-2.noach.rpm and php-gd-5.2.6-2.i380.rpm packages are installed and the following commands entered:

```
pear install Image_Color

pear install Log

pear install Numbers_Roman

pear install http://pear.php.net/get/Numbers_Words-0.15.0.tgz

pear install http://pear.php.net/get/Image_Graph-0.3dev4.tgz
```

2. Automation of Attack Response

The DDoS mitigation technique introduced by this thesis research is a reactive technique. To achieve the automated response, the Intrusion Detection System should not only *log* events, but also *react* to the attack attempts. Such a behavioral enhancement turns the IDS (detection only) into an Intrusion Detection and Prevention (IDP) solution.

Snort® provides the capability to analyze data and take action based on the results. Techniques used to take action can be written in one's own custom script, using an available plug-in, by writing one's own plug-in. As Puri stated in his study:

> After thorough research, it was found that a Snort has been extended with an output plug-in that notifies the SnortSam agent of blocking requests on a rule basis. SnortSam, developed by Mr. Frank Knobe (www.snortsam.net) is an intelligent agent that allows Snort to block connections by configuring firewalls or routers. SnortSam requires the Snort rule to be modified. The biggest advantage of this SnortSam agent is that it is built on the client-agent-based concept. SnortSam runs as an independent process and does not increase the workload of Snort.

For the above advantages and in order to produce comparable results with Puri's research, the SnortSam plug-in program is chosen to achieve the IDS's automatic reaction.

3. Install SnortSam

The SnortSam is installed in accordance with SnortSam's installation guide [27].

The source file (snortsam-src-2.60.tar.gz) has been downloaded from the SnortSam web site at http://www.snortsam.net and installed by issuing the following commands:

```
tar zxvf snortsam-src-2.60.tar.gz

cd snortsam
```

```
chmod +x makesnortsam.sh

./makesnortsam.sh
```

Since SnortSam is compiled, the binary is copied into the folder `/usr/local/bin`.

The next step is to add the SnortSam plug-in into Snort®. The snortsam-patch.tar.gz file from the SnortSam web site is downloaded and the following commands are entered in order to install it:

```
tar zxvf snortsam-patch.tar.gz

chmod +x patchsnort.sh

./patchsnort.sh /snort-2.6.1.3/
```

Then Snort® is configured with the commands previously shown.

After installing the SnortSam module, the snortsam.conf file, located under the /snortsam/conf directory, is configured. The file is edited and the following lines, after all the commented lines are added:

```
accept 192.168.3.0/24

accept localhost

logfile /var/log/snortsam.log

daemon

cisconullroute 192.168.10.5 vordos vordos

cisconullroute 192.168.10.3 vordos vordos

cisconullroute 192.168.10.4 vordos vordos
```

The above configuration tells the SnortSam client to accept the connections from the local host as well as 192.168.3.0/24 (the IDS's subnetwork). The logfile option tells it where to log files. Finally, the commands "`cisconullroute 192.168.10.5 vordos vordos`" ,"`cisconullroute 192.168.10.3 vordos vordos`" and "`cisconullroute 192.168.10.4`

vordos vordos" tells the client to use the cisconullroute plug-in three times with a different router's IP address each time; 192.168.10.5, 192.168.10.3 and 192.168.10.3 are the IP addresses of the routers where the SnortSam module will log in. The first "vordos" is the login password for the telnet session and the second "vordos" is the password to enter the configuration mode of the Cisco router.

The next step is to reconfigure the /etc/snort/snort.conf file. The output plug-in needs to be added so that Snort® can send the block request of the destination IP address. The following command is added in the snort.conf file.

```
output alert_fwsam: 127.0.0.1
```

That command told Snort® to send the blocking request to the local machine. The IP address 127.0.0.1 indicates that the SnortSam module is configured on the same machine where Snort is configured.

Snort allows users to write their own rules as per organizational requirements. By default, all the Snort rules are found in the /etc/snort/rules directory. The rules folder contains a file named "local.rules" through which the user can add customized rules. The following rule is added to this file to invoke a blocking of the destination IP address on the Cisco routers.

```
alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg: "ICMP Denial of Service Test"; itype: 8; classtype: misc-activity; threshold: type both, track by_dst, count 100, seconds 10; sid: 1000001; rev: 1 ; fwsam: dst, 20 minutes;)
```

In general, this rule will look for a minimum of 100 ICMP echo request packets within 10 seconds before generating an alert and then ignore the rest of the packets. The option "fwsam: dst, 20 minutes;" at the end of the rule body told Snort® to invoke a block of 20 minutes on the destination address via the SnortSam module whenever the above rule fired. The alert is presented on the BASE interface with the message "ICMP Denial of Service Test", as in the following figure:

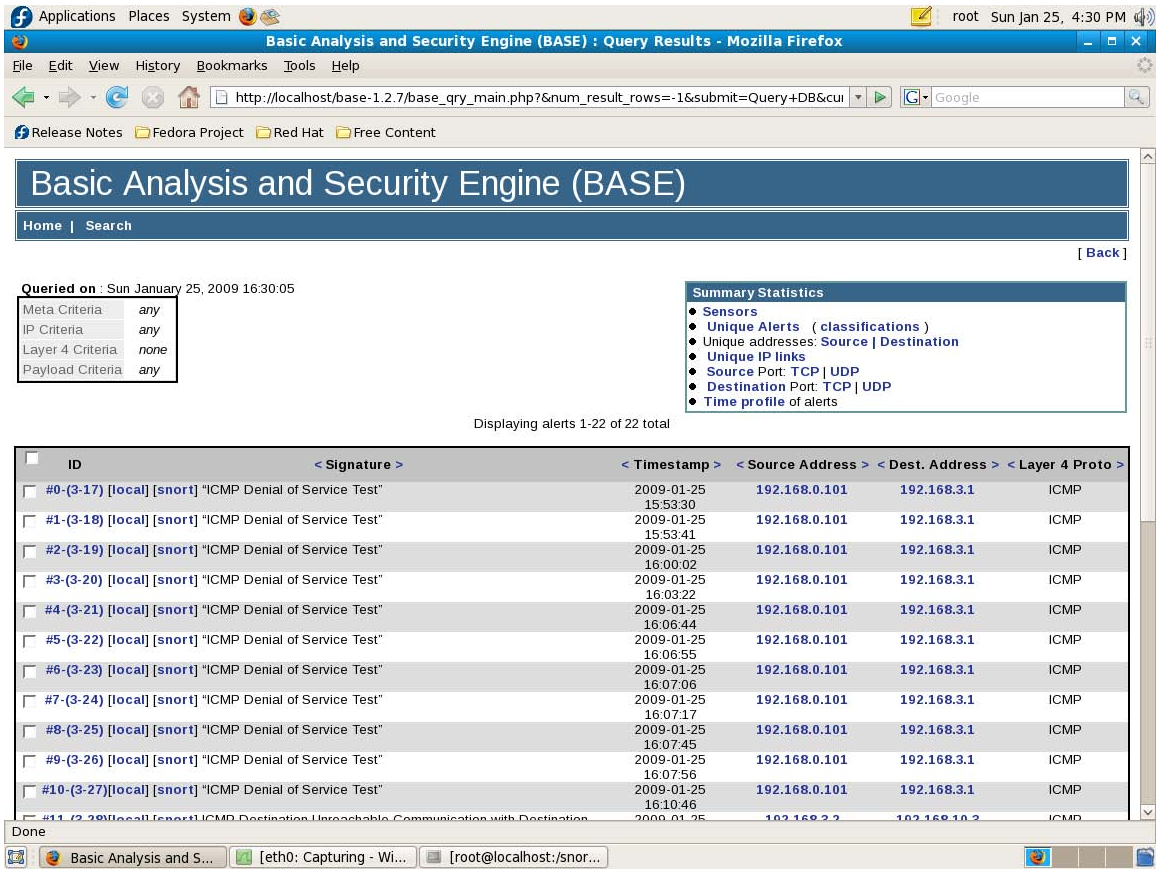


Figure 19. Successful activation of custom rule

Further details for the above rule are provided in Puri's thesis Appendix G [3].

After modifying the above files, we give the following command to restart Snort.

```
service snortd restart
```

Then the following command is entered:

```
./snortsam conf/snortsam.conf
```

From this point forward the SnortSam is running and listening for alerts from Snort®.

4. Modify ing the Plug-in's Source Code

The ssp_cisco_nullroute.c is a C file and it is one of SnortSam's plug-in programs. Its original purpose is to perform null routing, in a BGP BHR technique, by doing the following three things:

- Logs on the trigger router via telnet.
- Issues a command to enter the "null-route."
- When the time interval of blocking expires, it removes the added route to null0.

This program is found to be close to this research's intentions and so the author decided to use it with the following modifications. At the beginning its original command "`ip route %s 255.255.255.255 null 0\r`" is replaced with the following 3 new commands:

```
ip route %s 255.255.255.255 ethernet1/0 \r
ip route %s 255.255.255.255 tunnel11 \r
ip route %s 255.255.255.255 tunnel12 \r
```

Only one of the above commands is executed in each telnet session and the right choice between them is based on the provided router's ip address by the snortsam.conf file, through an "`if`" command.

Furthermore, the original program did not terminate each time the telnet session and that caused delays to the response time.

Finally, each command's number of characters is reduced to the minimum accepted from a Cisco router, in order to further reduce each telnet session's duration.

The modified program now does the following.

- Logs on the cleaning center's LER router via telnet.

- Issues the command “`ip route 192.168.3.1 255.255.255.255 ethernet1/0\r.`”
- Logs on the LER2 router via telnet.
- Issues the command “`ip route 192.168.3.1 255.255.255.255 tunnel12.`”
- Logs on the LER3 router via telnet.
- Issues the command “`ip route 192.168.3.1 255.255.255.255 tunnel11.`”
- When the time interval of blocking expires, it removes the previously added static routes.

The modified C file is attached as Appendix B.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. TESTI NG—RESULTS—ANALYSIS

A. CHAPTER OVERVIEW

This chapter presents the experimental results from testing the test-bed network against a series of manufactured DDoS attacks of different intensities. The second section of this chapter described how the network operated before the attacks and the typical sequence of events in its response to one such attack. The third section described the performance metrics used in this thesis. The fourth section presented the analysis of the collected timing results from all the attacks. Finally, in the fifth section, the performance results are compared with those reported for BGP BHR.

B. TESING

1. Before the Attack

Before the attack, the network was under normal operation. The LER2 forwarded traffic to the target host through MPLS tunnel 13 and LER3 through tunnel 10. Each of these tunnels was configured with a bandwidth of 4.8 Mbps. The reverse traffic from the target host to LER2 was transported through tunnel 15 and to LER3 through tunnel 16.

The scenario is depicted in Figure 20.

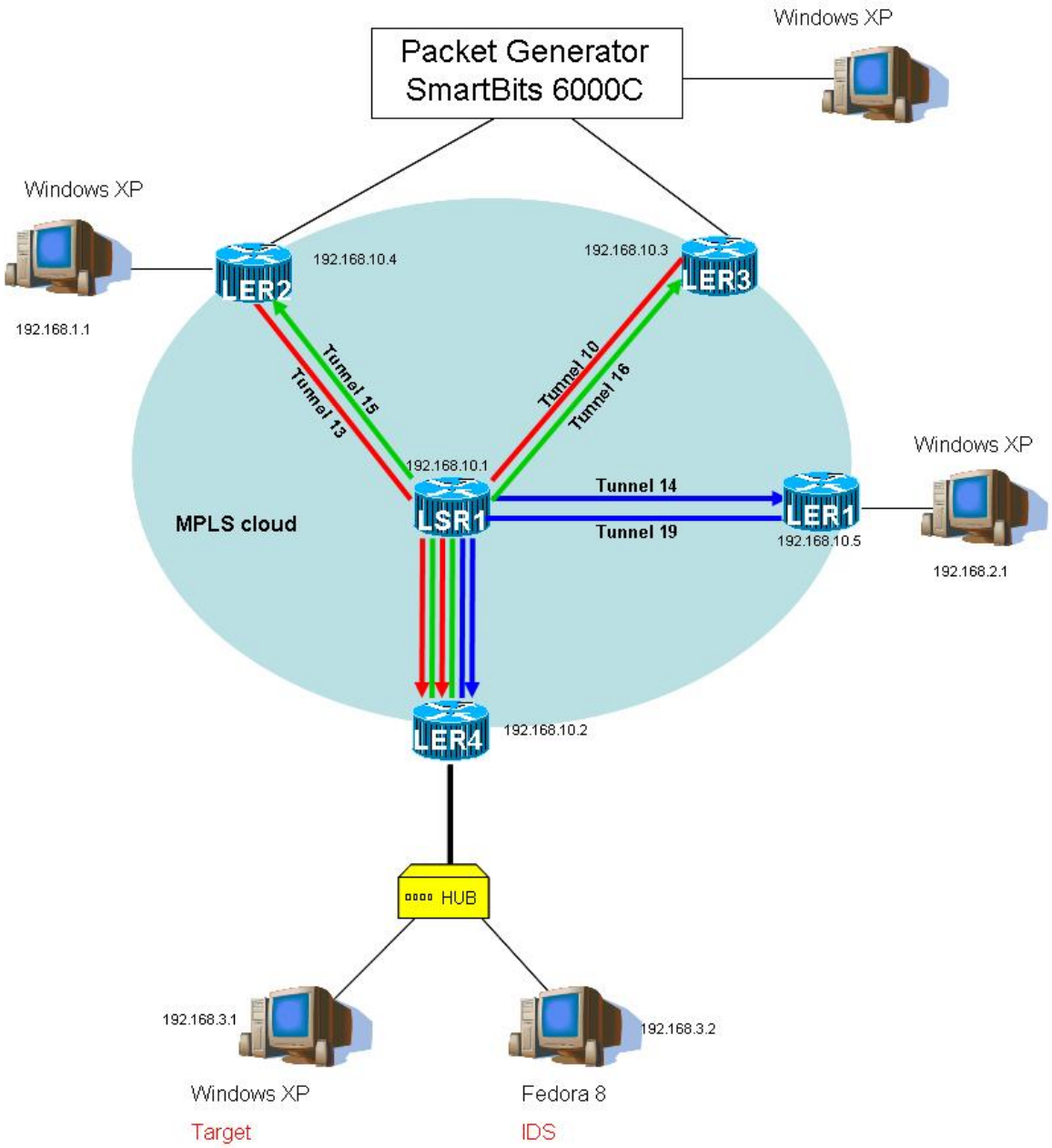


Figure 20. Test-bed before the attack

The forwarding tables of routers LER1, LER2 and LER3 before the attack were as in the following three Tables 4 to 6. Bold letters show the default routes for the target's network before the attack.

Forwarding Table of LER1 Before the Attack
LER1#show ip route Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2 E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2 ia - IS-IS inter area, * - candidate default, U - per-user static route o - ODR, P - periodic downloaded static route Gateway of last resort is not set 192.168.10.0/32 is subnetted, 5 subnets O 192.168.10.2 [110/21] via 0.0.0.0, 00:07:34, Tunnel19 O 192.168.10.3 [110/21] via 0.0.0.0, 00:07:34, Tunnel18 O 192.168.10.1 [110/11] via 192.168.7.1, 00:07:34, Ethernet1/1 O 192.168.10.4 [110/21] via 0.0.0.0, 00:07:34, Tunnel17 C 192.168.10.5 is directly connected, Loopback0 O 192.168.4.0/24 [110/20] via 192.168.7.1, 00:07:34, Ethernet1/1 O 192.168.5.0/24 [110/20] via 192.168.7.1, 00:07:34, Ethernet1/1 O 192.168.6.0/24 [110/20] via 192.168.7.1, 00:07:34, Ethernet1/1 C 192.168.7.0/24 is directly connected, Ethernet1/1 192.168.0.0/30 is subnetted, 1 subnets O 192.168.0.100 [110/30] via 0.0.0.0, 00:07:34, Tunnel17 [110/30] via 0.0.0.0, 00:07:34, Tunnel18 O 192.168.1.0/24 [110/30] via 0.0.0.0, 00:07:34, Tunnel17 C 192.168.2.0/24 is directly connected, Ethernet1/0 **O 192.168.3.0/24 [110/30] via 0.0.0.0, 00:07:36, Tunnel19**

Table 4. LER1's Forwarding Table Before the Attack

Forwarding Table of LER2 Before the Attack
LER2#show ip route Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2 E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2 ia - IS-IS inter area, * - candidate default, U - per-user static route o - ODR, P - periodic downloaded static route Gateway of last resort is not set 192.168.10.0/32 is subnetted, 5 subnets O 192.168.10.2 [110/21] via 0.0.0.0, 00:08:23, Tunnel13 O 192.168.10.3 [110/21] via 192.168.4.1, 00:08:23, Ethernet0/1 O 192.168.10.1 [110/11] via 192.168.4.1, 00:08:23, Ethernet0/1 C 192.168.10.4 is directly connected, Loopback0 O 192.168.10.5 [110/21] via 0.0.0.0, 00:08:23, Tunnel12 C 192.168.4.0/24 is directly connected, Ethernet0/1 O 192.168.5.0/24 [110/20] via 192.168.4.1, 00:08:23, Ethernet0/1 O 192.168.6.0/24 [110/20] via 192.168.4.1, 00:08:23, Ethernet0/1 O 192.168.7.0/24 [110/20] via 192.168.4.1, 00:08:23, Ethernet0/1 192.168.0.0/30 is subnetted, 1 subnets C 192.168.0.100 is directly connected, Ethernet1/1 C 192.168.1.0/24 is directly connected, Ethernet0/0 O 192.168.2.0/24 [110/30] via 0.0.0.0, 00:08:23, Tunnel12 **O 192.168.3.0/24 [110/30] via 0.0.0.0, 00:08:24, Tunnel13**

Table 5. LER2's Forwarding Table Before the Attack

Forwarding Table of LER3 Before the Attack
LER3#show ip route Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2 E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2 ia - IS-IS inter area, * - candidate default, U - per-user static route o - ODR, P - periodic downloaded static route Gateway of last resort is not set 192.168.10.0/32 is subnetted, 5 subnets O 192.168.10.2 [110/21] via 0.0.0.0, 00:06:02, Tunnel10 C 192.168.10.3 is directly connected, Loopback0 O 192.168.10.1 [110/11] via 192.168.5.1, 00:06:02, Ethernet0/1 O 192.168.10.4 [110/21] via 192.168.5.1, 00:06:02, Ethernet0/1 O 192.168.10.5 [110/21] via 0.0.0.0, 00:06:02, Tunnel11 O 192.168.4.0/24 [110/20] via 192.168.5.1, 00:06:02, Ethernet0/1 C 192.168.5.0/24 is directly connected, Ethernet0/1 O 192.168.6.0/24 [110/20] via 192.168.5.1, 00:06:02, Ethernet0/1 O 192.168.7.0/24 [110/20] via 192.168.5.1, 00:06:02, Ethernet0/1 192.168.0.0/30 is subnetted, 1 subnets C 192.168.0.100 is directly connected, Ethernet1/1 O 192.168.1.0/24 [110/30] via 192.168.5.1, 00:06:02, Ethernet0/1 O 192.168.2.0/24 [110/30] via 0.0.0.0, 00:06:02, Tunnel11 **O 192.168.3.0/24 [110/30] via 0.0.0.0, 00:06:02, Tunnel10**

Table 6. LER3's Forwarding Table Before the Attack

2. During the Attack

As stated in Chapter III, the selected type of attack was an ICMP flood attack launched from the SmartBits 6000C system. Different attack flows were created with the SmartBits application. A total of eleven different attack flows were evaluated, each with a different traffic intensity, starting from a relatively small number of frames per second (fps) up to the maximum capability of the packet generator for the specific connections created. Each flow was divided equally into two parts so that about a half of the total traffic would pass through each of the two border routers. For the final attack flow, the maximum bit- rate of the packet generator in every port was used. These flows are presented in Table 7.

Attack Flow #	Total Frame Rate (fps)	Total Bit Rate (Mbps)
1	1688	1.84
2	3376	3.34
3	5066	5.02
4	6756	6.7
5	8272	8.36
6	8444	8.53
7	10134	10.04
8	11820	11.72
9	13512	13.38
10	15202	15.08
11	16890	16.76

Table 7. Attack Flows

Flow #5 was the maximum flow under which the CPU load of every router was below 80 percent in the test-bed. Above this level, LSR1 reached a state of CPU overload and its behavior became very unstable. Because of that, it was assumed that Flows #6 through #11 simulate a heavy DDoS attack for networks where the limiting factor is the router CPU load.

The sequence of events triggered by an attack with Flow #6 is presented here. After the attack was started, the malicious packets started to show on the capture window of the Wireshark application running on the target host. Figure 21 shows a snapshot of that capture window. The packets of the attack flow with their source IP address 192.168.1.101 can be identified.

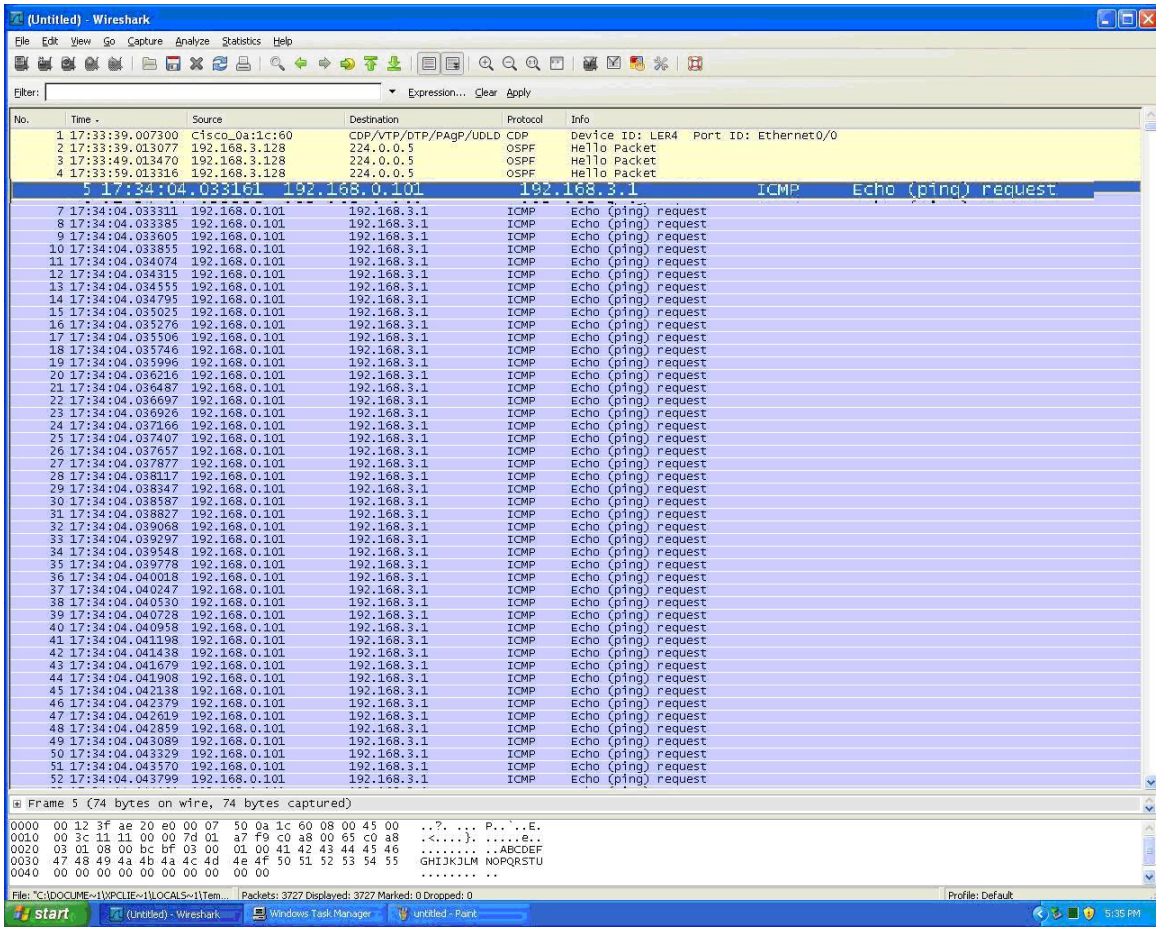


Figure 21. Initiation of attack captured by target host's Wireshark application

As discussed in Chapter III, the IDS/automation host was connected to the same network as the target host through a hub. Once the IDS detected the attack, it invoked the customized "ssp_cisco_nullroute.c" program. At the beginning the IDS/automation host (with IP address 192.168.3.2) started a telnet session with LER1 (with IP address 192.168.10.5) and added a static route for transporting the attack traffic to the cleaning center with the router configuration command "ip route 192.168.3.1 255.255.255.255 ethernet 1/0" as captured in Figure 22. The telnet session was initiated 0.033 seconds after the attack was launched, as shown in Figure 23.

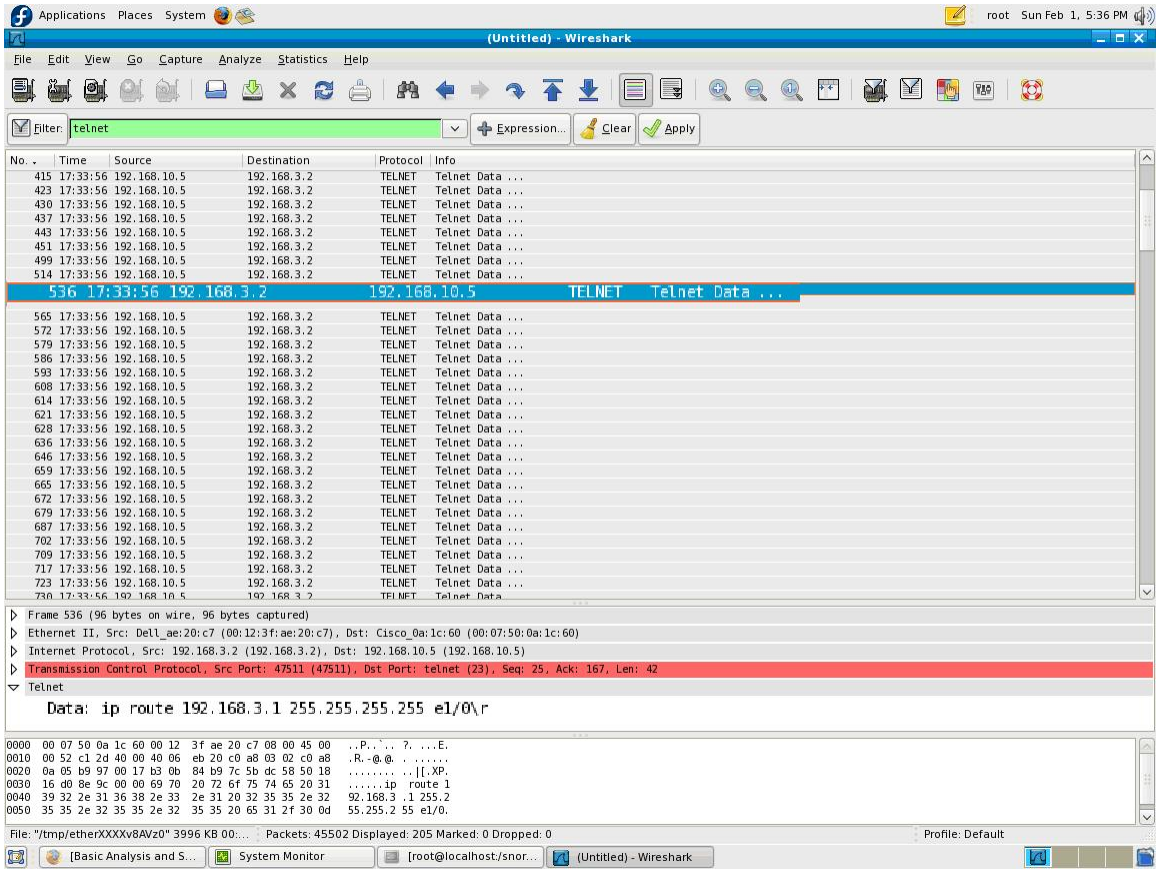


Figure 22. Telnet commands used to add the static route to LER1

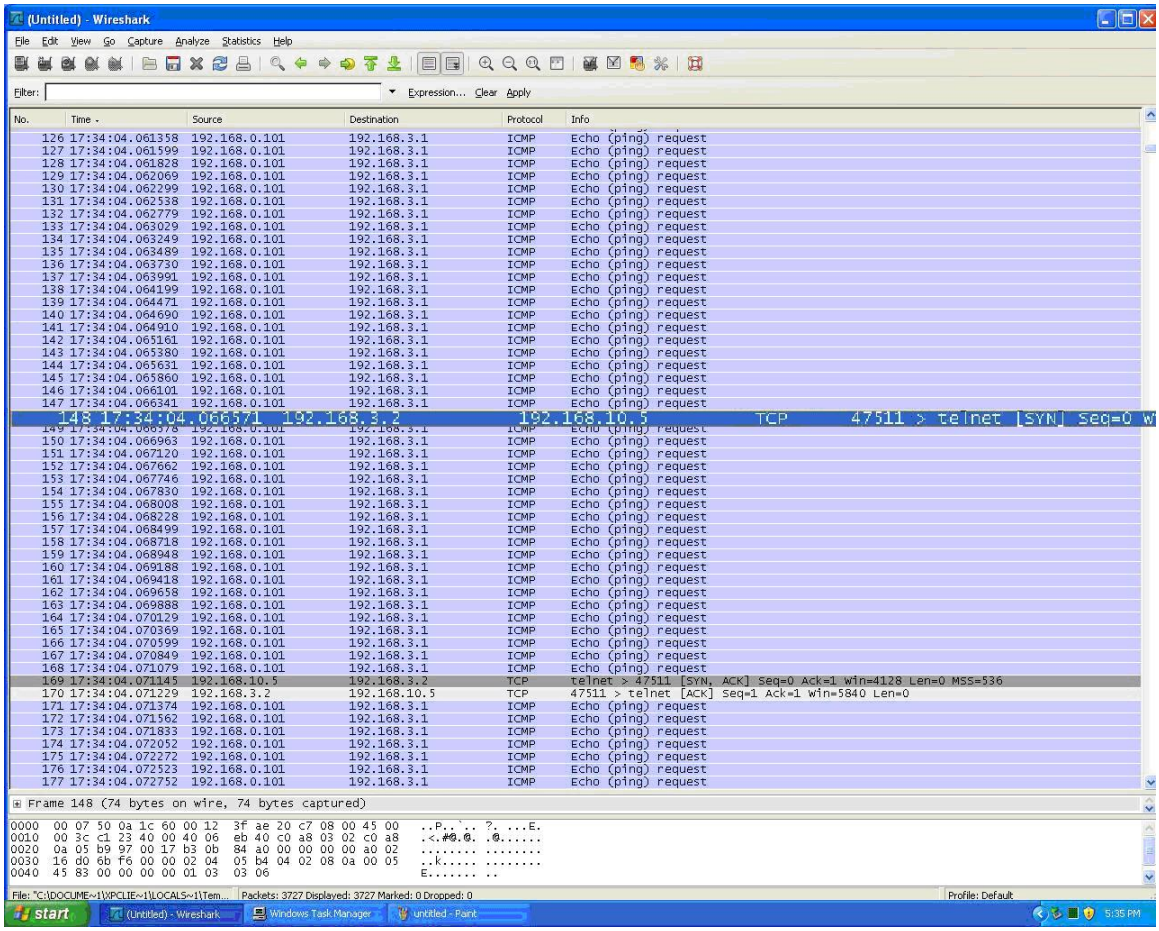


Figure 23. Snapshot of Wireshark capture window showing the system's first response

To redirect the attack traffic from the two border routers to LER1, the IDS/automation host then launched two more telnet sessions: one for logging on the border router LER2 (with IP address 192.168.10.4) and adding a static route via the router configuration command "ip route 192.168.3.1 255.255.255.255 tunnel 12", and the other for remotely adding a static route to the border router LER3 (with IP address 192.168.10.3) via the router configuration command "ip route 192.168.3.1 255.255.255.255 tunnel 11". Figures 24 and 25 show the telnet commands for LER2 and LER3, respectively, captured by Wireshark when being sent from the IDS/automation host.

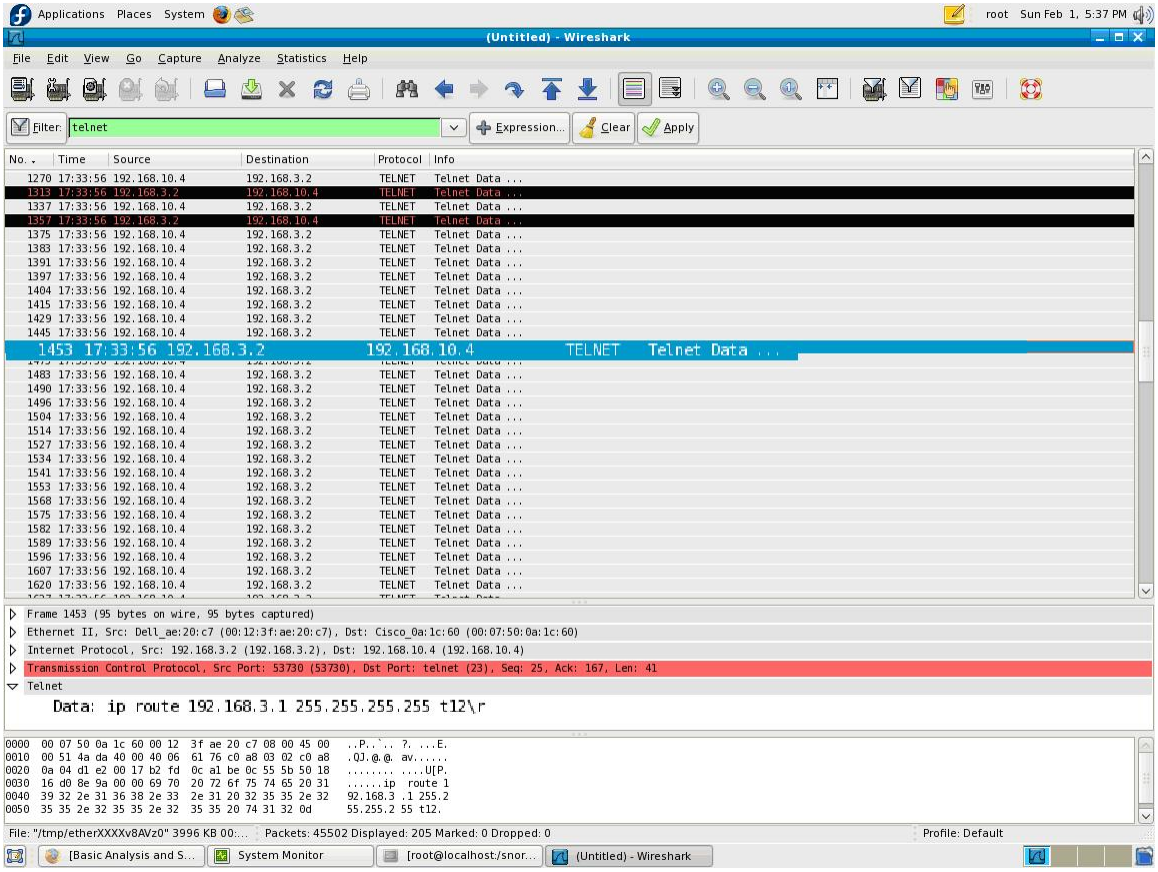


Figure 24. Telnet commands used to add the redirection route to LER2.

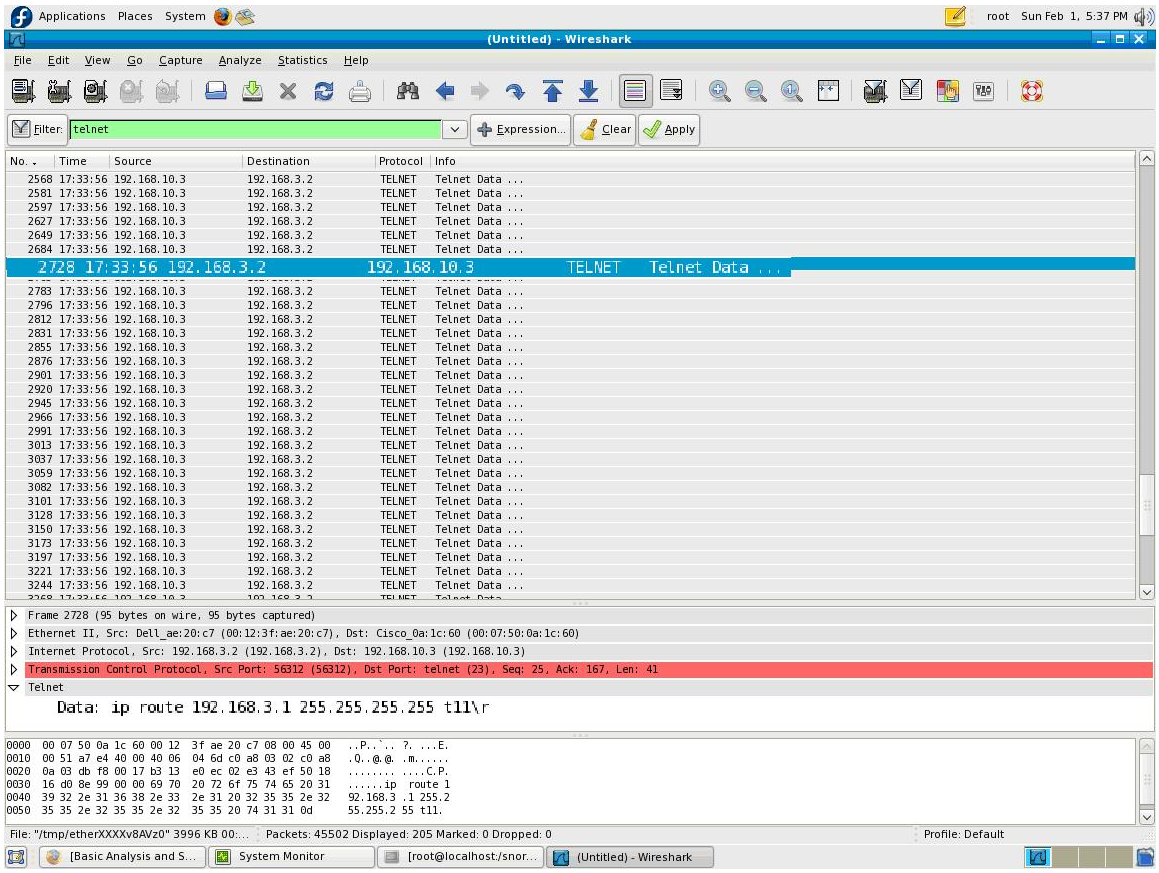


Figure 25. Telnet commands used to add the redirection route to LER3

All the static routes were added within a second. The last reception of a malicious packet on the target host happened 0.814 seconds after the reception of the first. Figure 26 shows the last received packet as captured by Wireshark running on the target host.

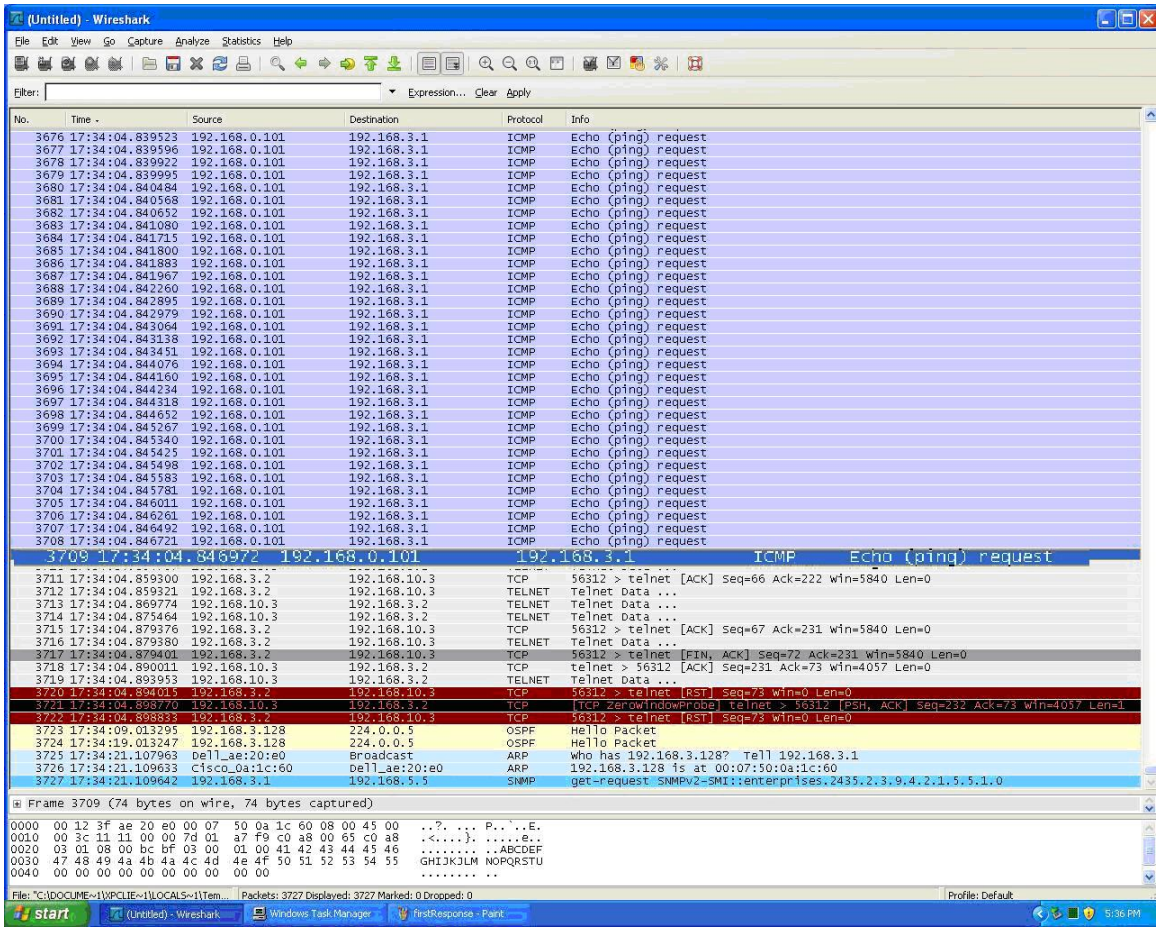


Figure 26. Snapshot of Wireshark capture window showing the attack's termination.

At this point, the DDoS attack against the target host was mitigated. All the malicious traffic was being redirected to the Windows XP machine simulating the network's cleaning center (i.e., the host with IP address 192.68.2.1 in Figure 20). This behavior can be seen from the following snapshot (Figure 27) captured by Wireshark running on the Windows XP machine.

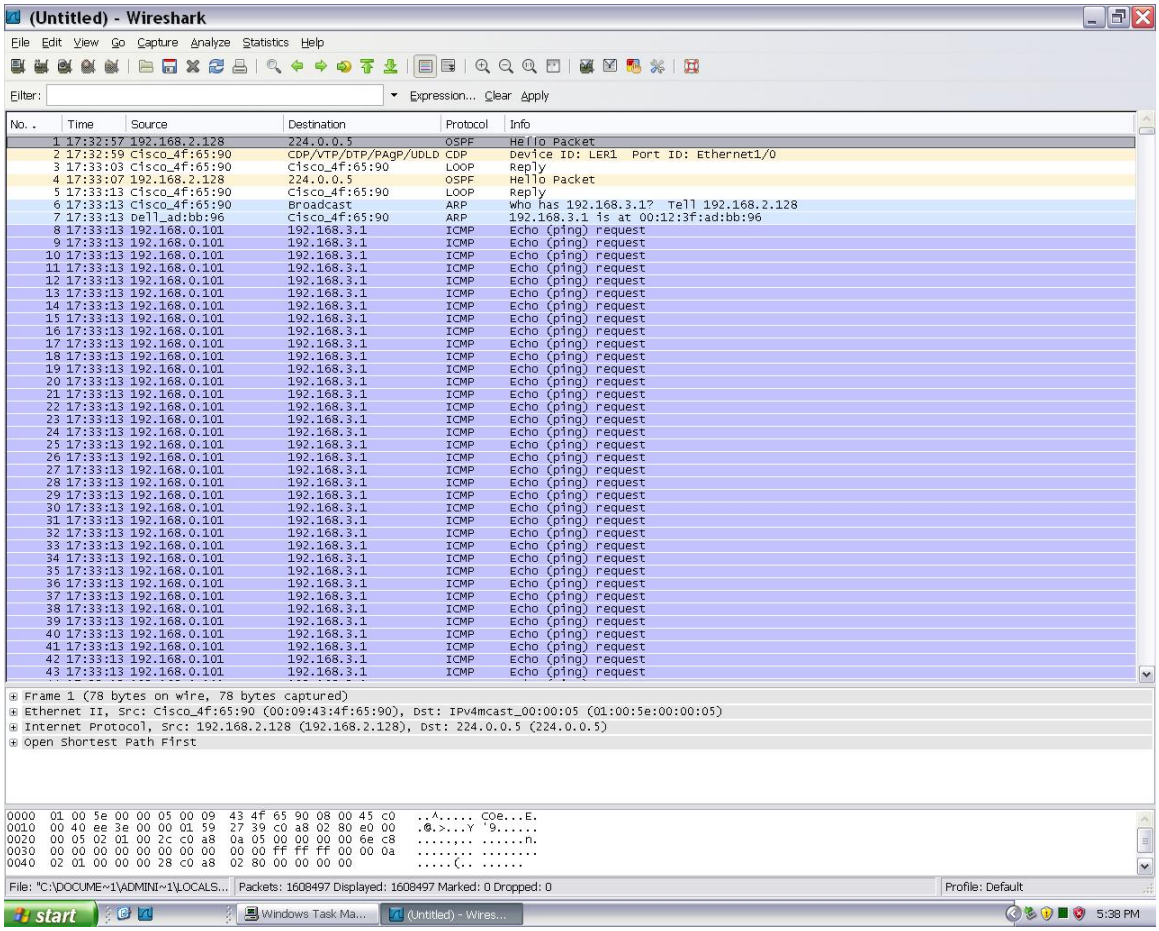


Figure 27. Snapshot from Wireshark running on the cleaning center host.

Tables 8-10 display the forwarding tables of routers LER1, LER2 and LER3 after the attack's mitigation. The entries with bold letters show the new static routes remotely added from the IDS/automation host.

Forwarding Table of LER1 After the Attack's Mitigation
LER1#show ip route Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2 E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2 ia - IS-IS inter area, * - candidate default, U - per-user static route o - ODR, P - periodic downloaded static route Gateway of last resort is not set 192.168.10.0/32 is subnetted, 5 subnets O 192.168.10.2 [110/21] via 0.0.0.0, 00:33:05, Tunnel19 O 192.168.10.3 [110/21] via 0.0.0.0, 00:33:05, Tunnel18 O 192.168.10.1 [110/11] via 192.168.7.1, 00:33:05, Ethernet1/1 O 192.168.10.4 [110/21] via 0.0.0.0, 00:33:05, Tunnel17 C 192.168.10.5 is directly connected, Loopback0 O 192.168.4.0/24 [110/20] via 192.168.7.1, 00:33:05, Ethernet1/1 O 192.168.5.0/24 [110/20] via 192.168.7.1, 00:33:05, Ethernet1/1 O 192.168.6.0/24 [110/20] via 192.168.7.1, 00:33:05, Ethernet1/1 C 192.168.7.0/24 is directly connected, Ethernet1/1 192.168.0.0/30 is subnetted, 1 subnets O 192.168.0.100 [110/30] via 0.0.0.0, 00:33:05, Tunnel17 [110/30] via 0.0.0.0, 00:33:05, Tunnel18 O 192.168.1.0/24 [110/30] via 0.0.0.0, 00:33:05, Tunnel17 C 192.168.2.0/24 is directly connected, Ethernet1/0 192.168.3.0/24 is variably subnetted, 2 subnets, 2 masks **S 192.168.3.1/32 is directly connected, Ethernet1/0** O 192.168.3.0/24 [110/30] via 0.0.0.0, 00:33:08, Tunnel19

Table 8. LER1's Forwarding Table after the mitigation of the attack

Forwarding Table of LER2 After the Attack's Mitigation
LER2#show ip route Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2 E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2 ia - IS-IS inter area, * - candidate default, U - per-user static route o - ODR, P - periodic downloaded static route Gateway of last resort is not set 192.168.10.0/32 is subnetted, 5 subnets O 192.168.10.2 [110/21] via 0.0.0.0, 00:35:31, Tunnel13 O 192.168.10.3 [110/21] via 192.168.4.1, 00:35:31, Ethernet0/1 O 192.168.10.1 [110/11] via 192.168.4.1, 00:35:31, Ethernet0/1 C 192.168.10.4 is directly connected, Loopback0 O 192.168.10.5 [110/21] via 0.0.0.0, 00:35:31, Tunnel12 C 192.168.4.0/24 is directly connected, Ethernet0/1 O 192.168.5.0/24 [110/20] via 192.168.4.1, 00:35:31, Ethernet0/1 O 192.168.6.0/24 [110/20] via 192.168.4.1, 00:35:31, Ethernet0/1 O 192.168.7.0/24 [110/20] via 192.168.4.1, 00:35:31, Ethernet0/1 192.168.0.0/30 is subnetted, 1 subnets C 192.168.0.100 is directly connected, Ethernet1/1 C 192.168.1.0/24 is directly connected, Ethernet0/0 O 192.168.2.0/24 [110/30] via 0.0.0.0, 00:35:31, Tunnel12 192.168.3.0/24 is variably subnetted, 2 subnets, 2 masks **S 192.168.3.1/32 is directly connected, Tunnel12** O 192.168.3.0/24 [110/30] via 0.0.0.0, 00:35:35, Tunnel13

Table 9. LER2's Forwarding Table after the mitigation of the attack

Forwarding Table of LER3 After the Attack's Mitigation

```
LER3#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

     192.168.10.0/32 is subnetted, 5 subnets
O       192.168.10.2 [110/21] via 0.0.0.0, 00:32:41, Tunnel10
C       192.168.10.3 is directly connected, Loopback0
O       192.168.10.1 [110/11] via 192.168.5.1, 00:32:41, Ethernet0/1
O       192.168.10.4 [110/21] via 192.168.5.1, 00:32:41, Ethernet0/1
O       192.168.10.5 [110/21] via 0.0.0.0, 00:32:41, Tunnel11
O    192.168.4.0/24 [110/20] via 192.168.5.1, 00:32:41, Ethernet0/1
C    192.168.5.0/24 is directly connected, Ethernet0/1
O    192.168.6.0/24 [110/20] via 192.168.5.1, 00:32:41, Ethernet0/1
O    192.168.7.0/24 [110/20] via 192.168.5.1, 00:32:41, Ethernet0/1
     192.168.0.0/30 is subnetted, 1 subnets
C       192.168.0.100 is directly connected, Ethernet1/1
O    192.168.1.0/24 [110/30] via 192.168.5.1, 00:32:41, Ethernet0/1
O    192.168.2.0/24 [110/30] via 0.0.0.0, 00:32:41, Tunnel11
     192.168.3.0/24 is variably subnetted, 2 subnets, 2 masks
S       192.168.3.1/32 is directly connected, Tunnel11
O       192.168.3.0/24 [110/30] via 0.0.0.0, 00:32:41, Tunnel10
```

Table 10. LER3's Forwarding Table after the mitigation of the attack

After the attack's redirection, LSR1 presented anomalies after about 1 minute. It started to lose connections with the rest of the routers. This unstable behavior was observed for attack Flows #6 to #11. As the traffic volume went higher, it took a shorter time for LSR1 to go down. When the attack traffic was the most intensive, i.e., 16.76 Mbps with attack Flow #11, LSR1's failure time was only 45 seconds. Further examinations revealed that the router's CPU load was more than 80% during those attacks, resulting in unstable behaviors. Figure 28 presents the error log messages from router LSR1 observed on the Windows XP machine.

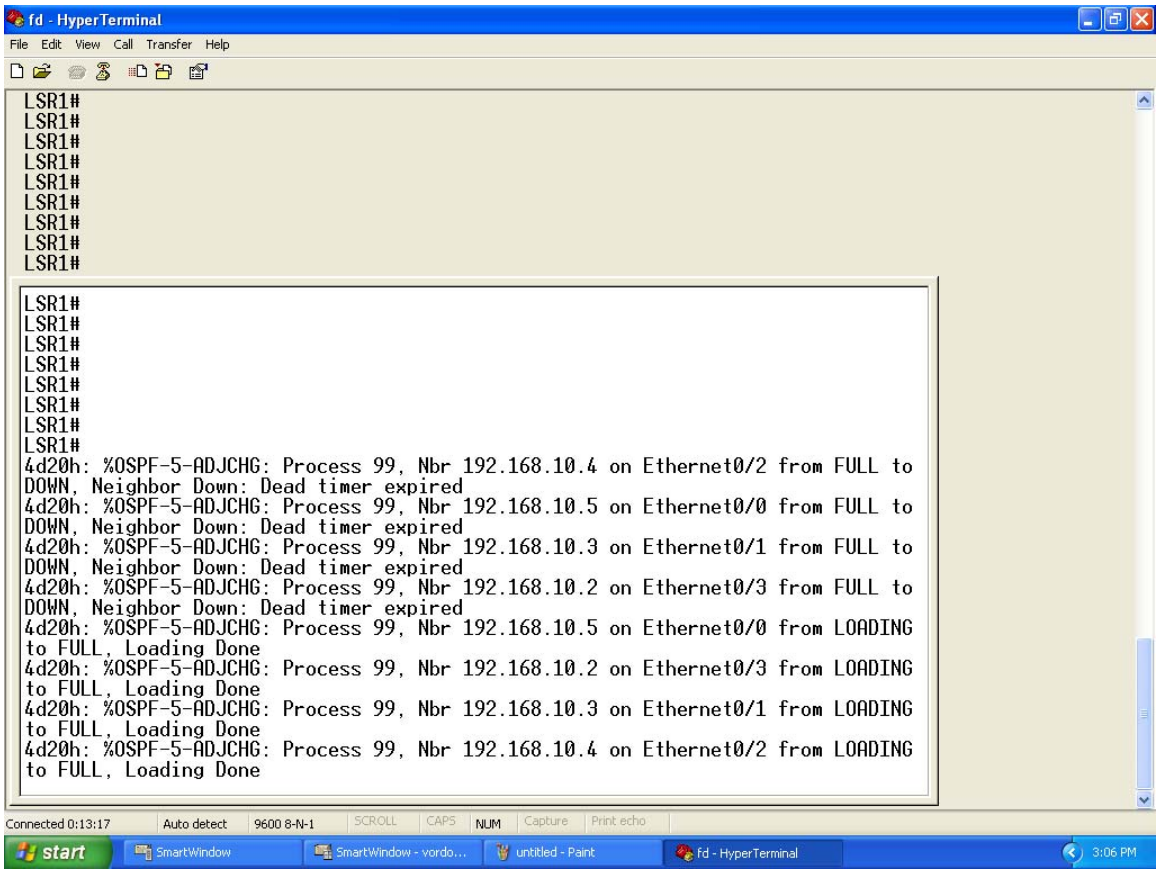


Figure 28. Router logs from LSR reporting unstable behaviors

3. Additional Test of Selective Unblocking

After the successful redirection of the attack traffic the part of the attack traffic coming in from router LER2 was manually stopped. A new telnet session from the IDS host with LER2 was started. The redirection route from LER2's forwarding table was removed through this session via the router configuration command "no ip route 192.168.3.1 255.255.255.255 tunnel12". Then, from the Windows XP machine (with IP address 192.168.1.1) connected to router LER2, the target host was pinged with the shell command "ping –t 192.168.3.1." These ping packets arrived at the target host. In other words, the traffic from LER2 was successfully returned to the normal path, as illustrated in Figure 29.

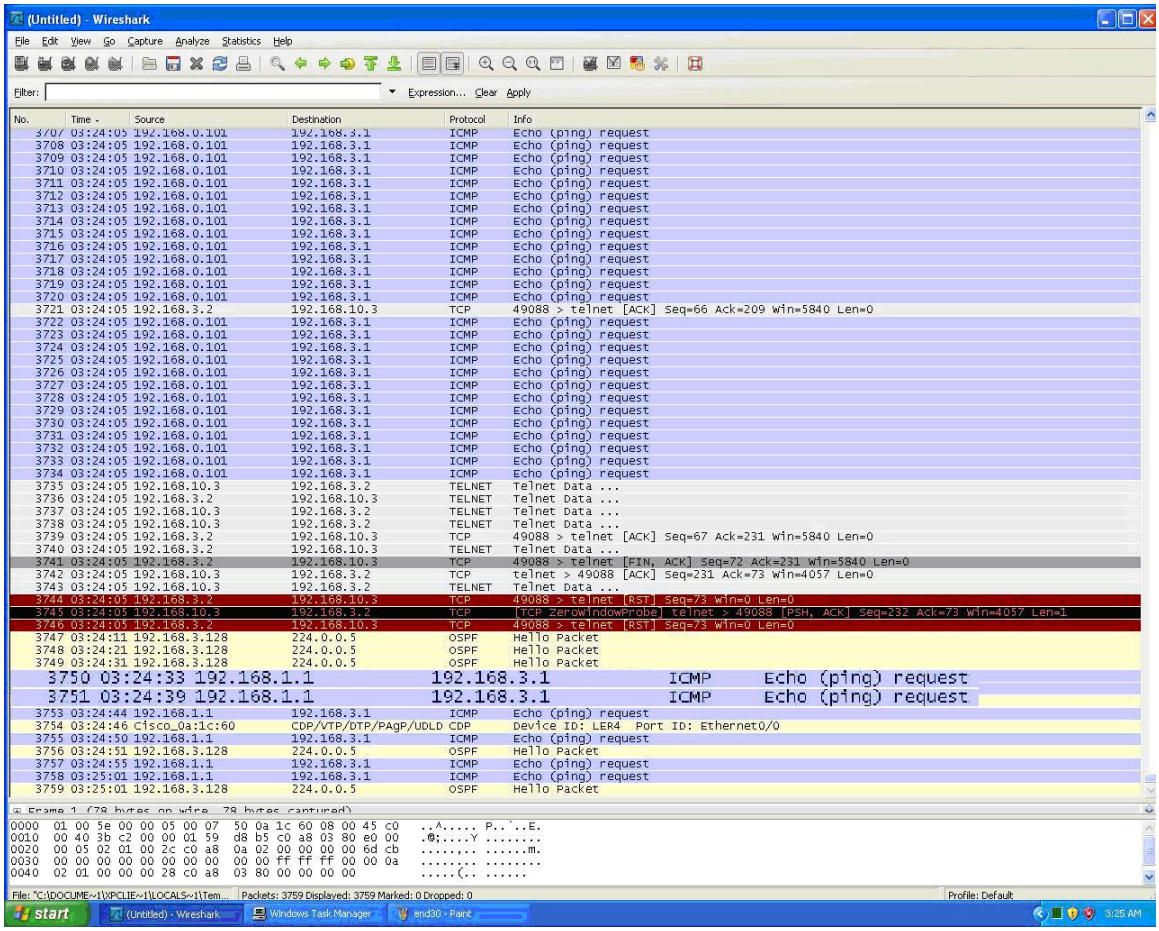


Figure 29. Snapshot from target's Wireshark after the selective unblocking on router LER2

C. PERFORMANCE METRICS

The main performance metric for this research is the overall response time of the network during an attack – which is referred to simply as "mitigation time" in the rest of this thesis. It is defined to be the time interval from the reception of the first malicious packet at the target host to the reception of the last malicious packet at the same host.

The IDS's first response time was also measured as the time interval between the reception of the first malicious packet at the target host and the transmission of the first redirection route (for LER1) from the IDS host.

To derive both performance metrics the Wireshark packet captures from the target host were used. The target and IDS hosts were connected to the same hub, hence, the Wireshark application running on the target host also captured packets from the IDS host. Wireshark gives time accuracy of 0.00001 second and this granularity was adequate for the needs of this research.

The last performance metric collected was the router CPU load. For this purpose the router configuration command “show processes cpu history" was used. This command displayed the router’s CPU load information for the last sixty seconds, one hour, and seventy two hours. Figure 30 presents an example output of this command.

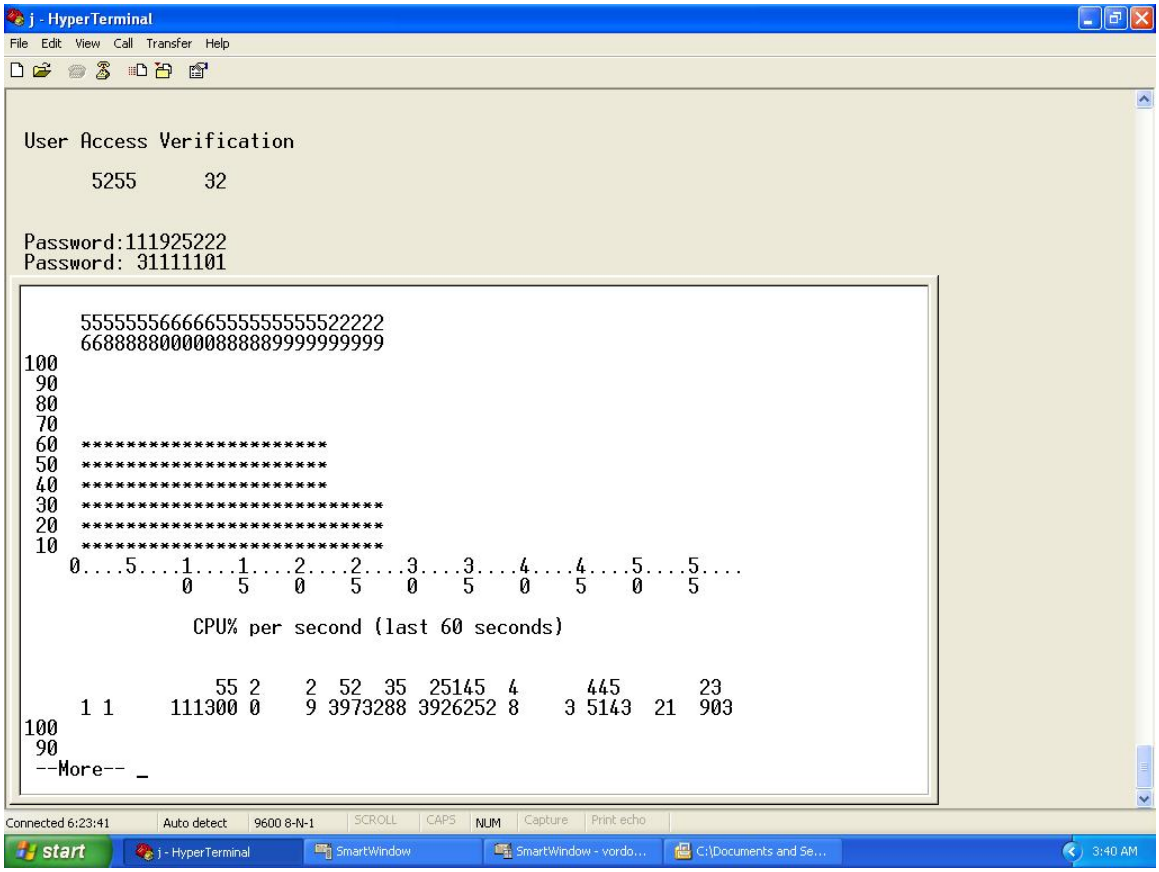


Figure 30. LER1 CPU load for attack flow 8.36 Mbps

D. EXPERIMENTAL RESULTS AND ANALYSIS

The results of the research are presented in this section. As noted in Chapter III, eleven different flows in the packet generator were specified. The first five flows were to simulate low-to-medium attack traffic for the specific routers used in the test-bed networks. The next five flows were to simulate high attack traffic. In order to achieve more accurate results, each experiment was run ten times, and then the average of the mitigation times and IDS's first response times were calculated. Figures 31 and 32 show the performance of the test-bed network under the different attack flows.

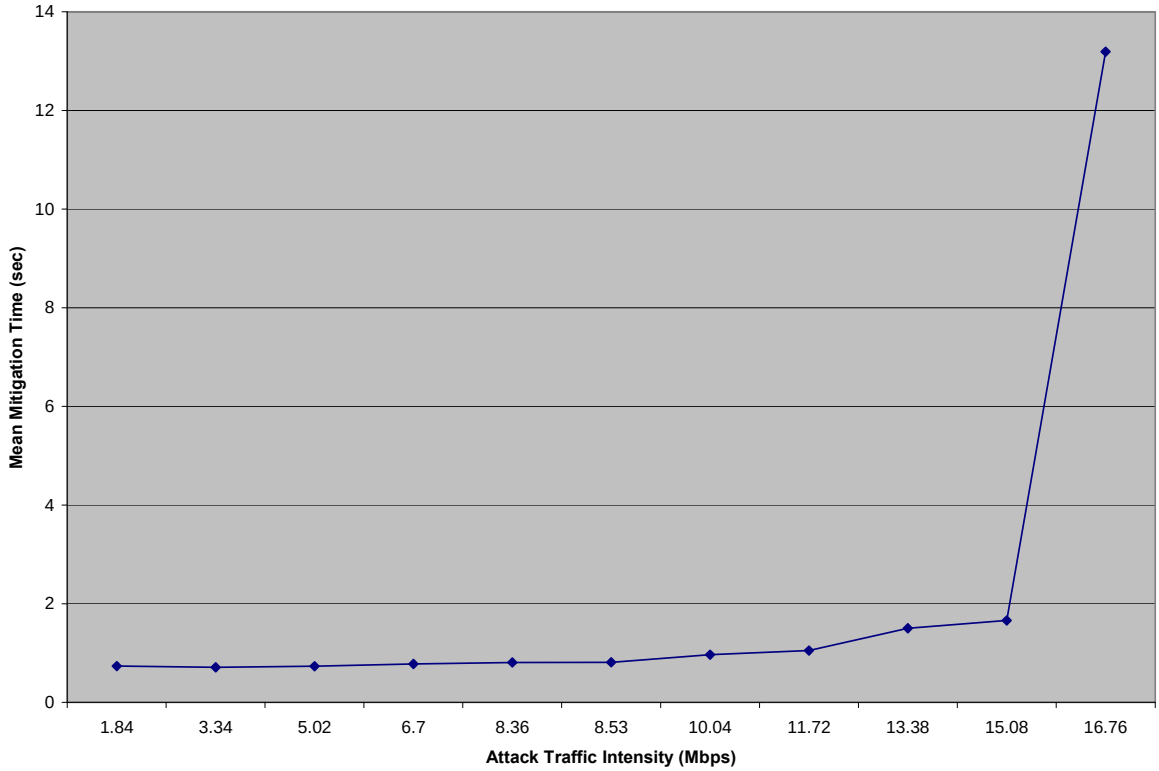


Figure 31. Mean mitigation time for different attack flows

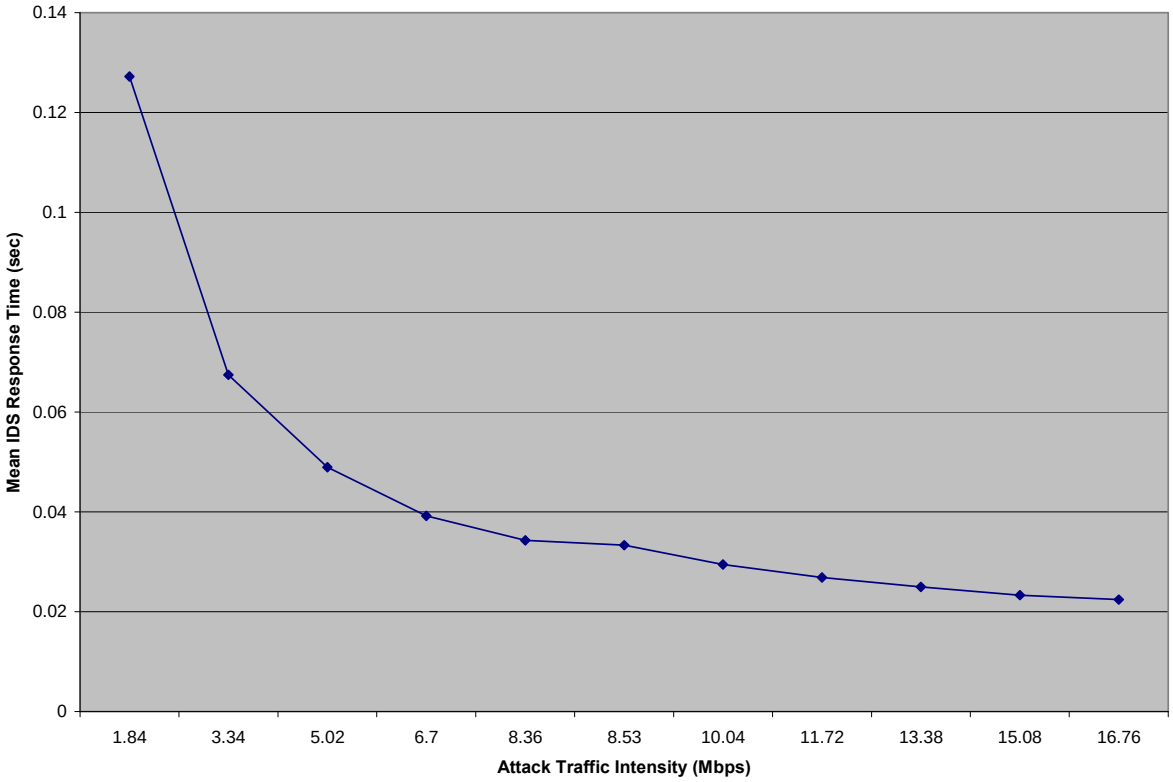


Figure 32. Mean time for IDS's first response for different attack flows

The numeric results are presented in Table 11.

Flow #	Attack traffic intensity (Mbps)	Mean Mitigation Time (sec)	Mean IDS First Response Time (sec)
1	1.84	0.73709	0.12718
2	3.34	0.71162	0.06746
3	5.02	0.73147	0.04893
4	6.7	0.77802	0.03918
5	8.36	0.80531	0.03431
6	8.53	0.81185	0.03332
7	10.04	0.96776	0.02943
8	11.72	1.05004	0.02683
9	13.38	1.50339	0.02498
10	15.08	1.65895	0.02333
11	16.76	13.18943*	0.02243

Table 11. Summary of timing data for different attack flows.

As can be extracted from these results, the system responds in a very short period of time under the first ten attack flows. The mitigation of DDoS attacks was achieved in less than 1 second for the first seven flows and less than 2 seconds for Flows # 8, 9 and 10. The IDS's first response time was decreased as the attack intensity was increased. This behavior was expected, since the Snort®'s rule used is fired after the reception of 100 ICMP echo request packets within 10 seconds, as referred to in Chapter III. Hence, the higher the attack flow , the faster the condition of this rule was met.

As already noted, for Flow #6 and higher, the CPU load of the core router LSR1 quickly exceeded 80%. Under this condition, LSR1 had an unstable behavior: first it output the error message for each of its interfaces "from FULL to DOWN, Neighbor Down: Dead timer expired," and very soon it wrote out another error message "from LOADING to FULL, Loading Done." The messages were due to the Open Shortest Path First (OSPF) protocol. The OSPF neighbors exchange "hello" packets at multicast address 224.0.0.5. If these packets are not delivered because of any Layer 2 issue, OSPF neighbors flap, resulting in the first error message [28]. In this case, some OSPF "hello" packets were dropped because of traffic congestion. The second message occurred because the router finally received a new "hello" packet from its neighbors and it loaded the interfaces again. Those problems had "gap" effects on the received traffic by the cleaning center as shown in Figure 33.

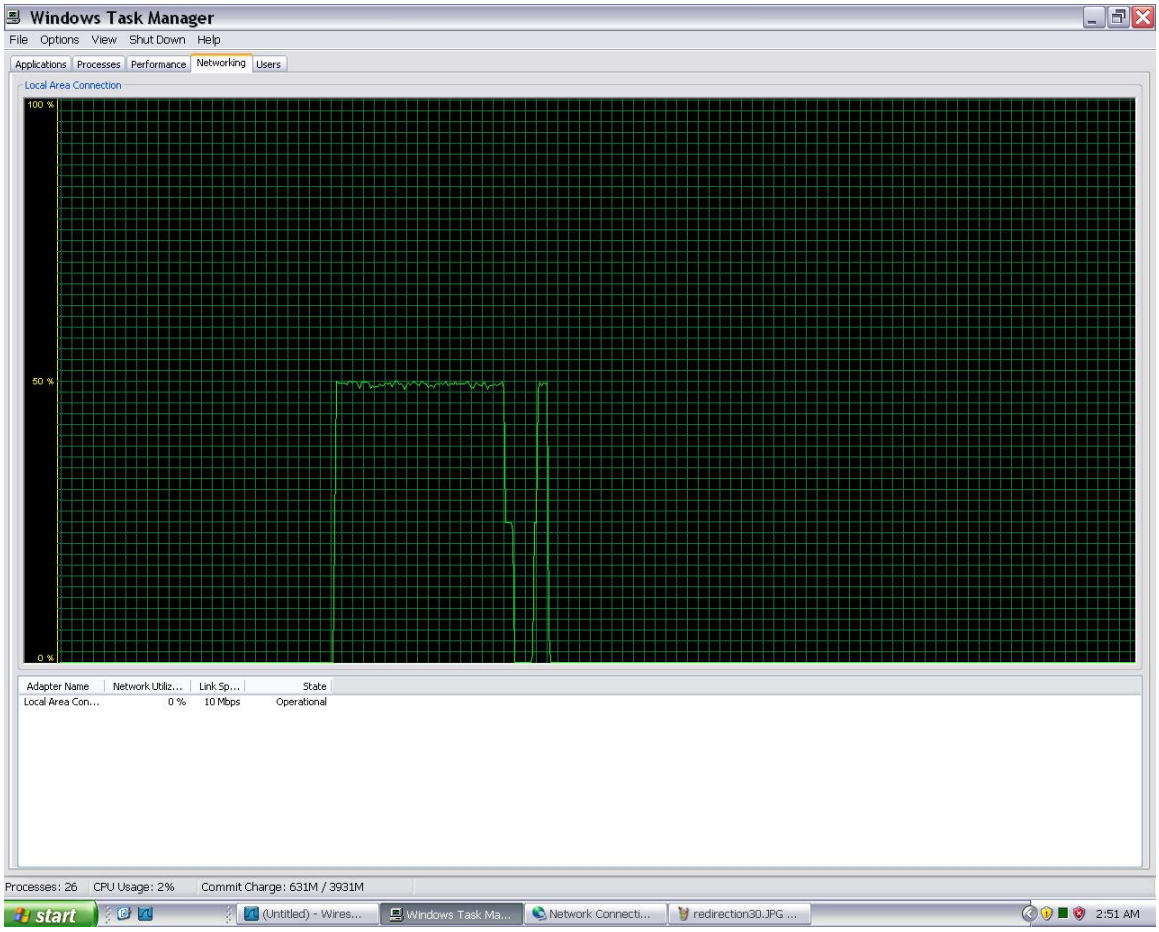


Figure 33. Traffic gaps at cleaning center host created by LSR1 failure due to heavy traffic

The network exhibited extensive anomalies under Flow #11. The anomalies had a large impact on the mitigation time. Highly varying values, from 4.6 seconds to 22 seconds, even though the measured IDS first response times were constant and very close to 0.022 seconds were obtained. In one case the system did not respond, even after 5 minutes. This problem was caused by high CPU load on router LSR1. (See Figure 37.) This high CPU load caused LSR1 to drop or delay the forwarding of the telnet packets used for mitigation. This router also became a traffic bottleneck even after redirection, because the two attack flows from routers LER2 and LER3 had to be consolidated into one MPLS-TE

tunnel at LSR1 first, on their way to LER1. Above Flow #6, consolidation was unattainable since the total attack flow rate was greater than the configured tunnel capacity of 10 Mbps.

Figures 34 to 37 show the CPU load measured on each network's router, except router LER4. The LER4 router (target's LER) had very low traffic, since mitigation was achieved in very short times.

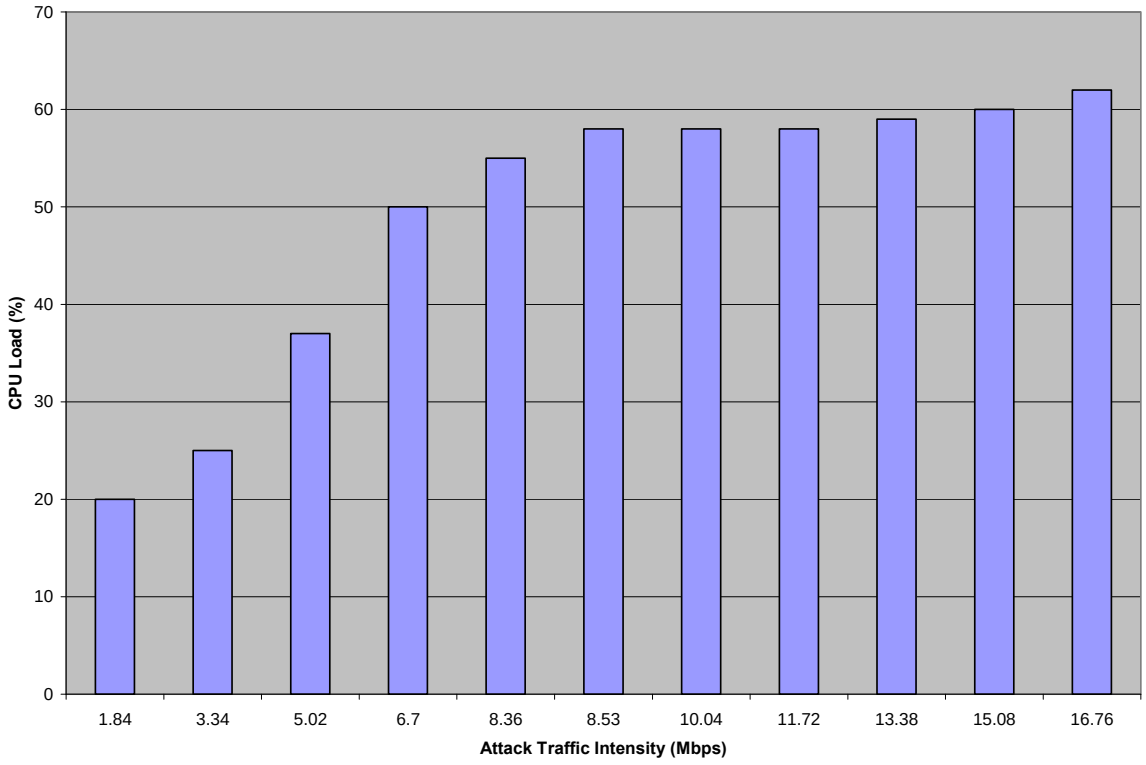


Figure 34. CPU load of LER1 (cleaning center)

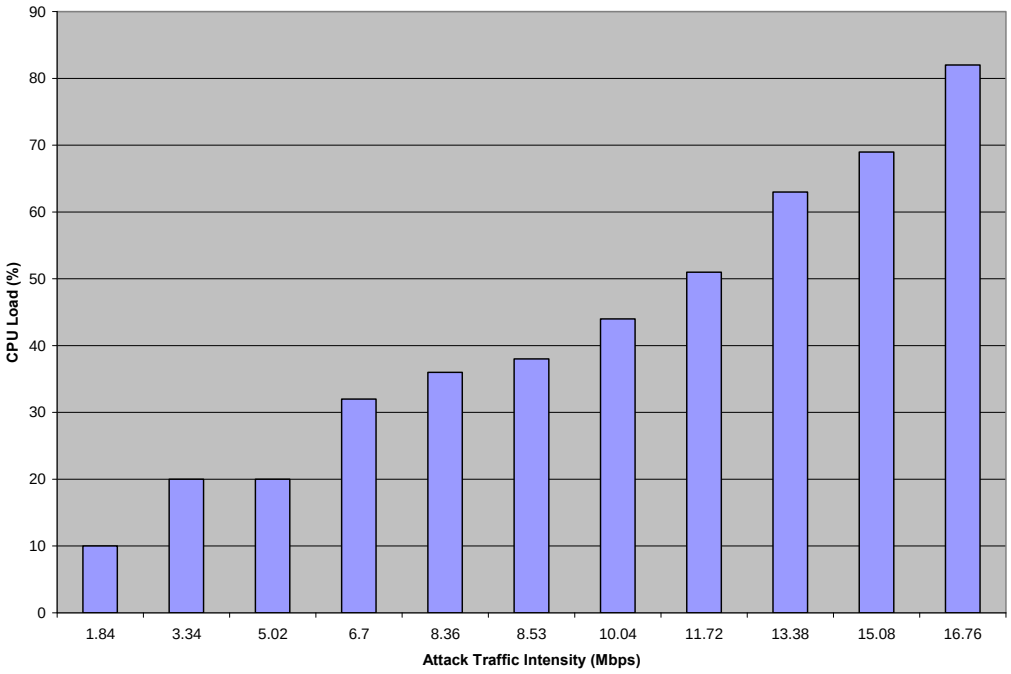


Figure 35. CPU load of LER2 (border router)

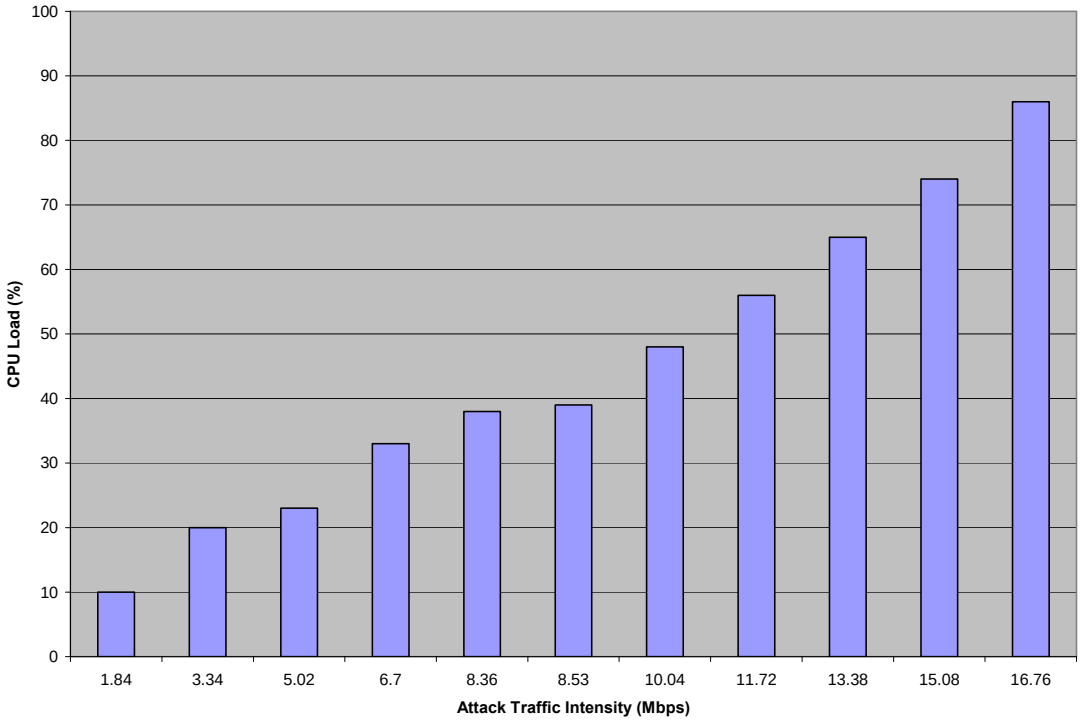


Figure 36. CPU load of LER3 (border router)

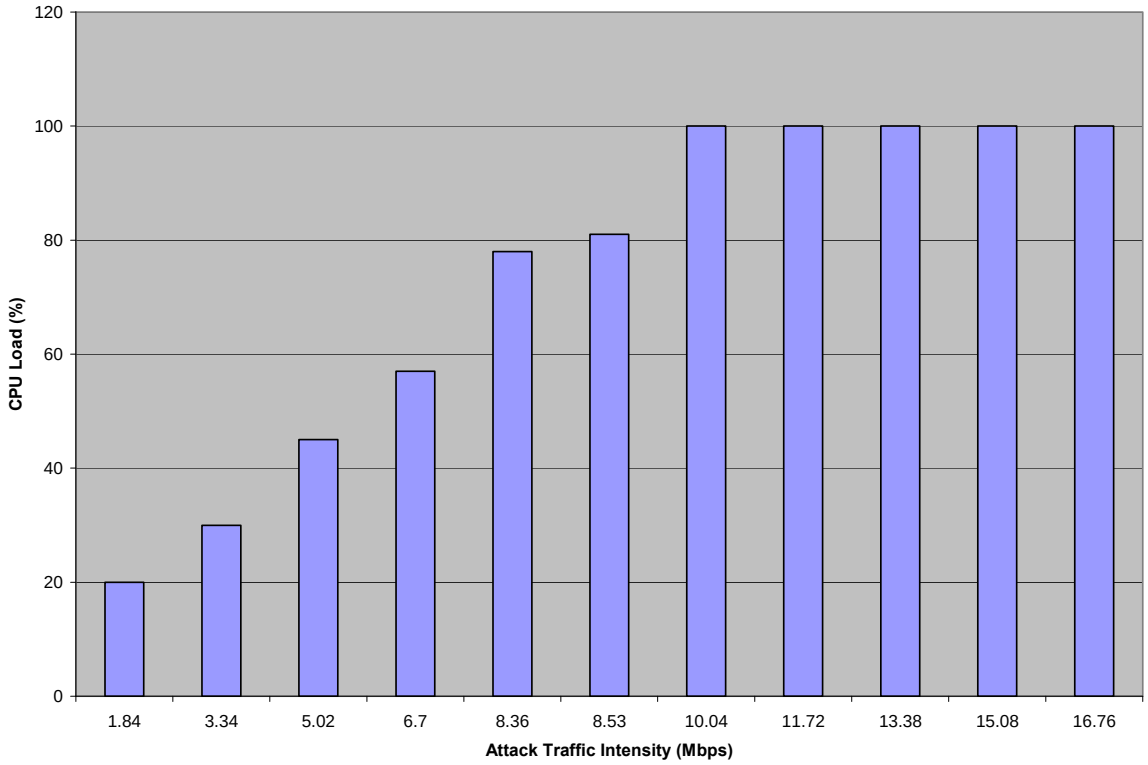


Figure 37. CPU load of LSR1 (core router)

The LER2 and LER3 routers had very similar CPU loads, as was expected, during all the tested attacks. They reached 80% CPU load only during the attacks with the maximum attack flow. Router LER1 never exceeded 65% CPU load. This can be explained by the already discussed bottleneck on LSR1.

E. COMPARISON BETWEEN MPLS-TE AND BGP BHR

Puri's study appears to be more related to this thesis research. His test-bed had almost the same topology. Furthermore, the IDS/automation system was almost identical. Unfortunately, Puri did not provide details about his testing methodology. He only alluded to the mitigation time in one of his conclusions, as follows:

> …once an attack was detected the system close to 20 seconds to mitigate the D/DoS attack. This time includes the telnet/SSH session initiated from the IDS to the trigger router, advertisement of the null route to all the border routers, and dropping all the malicious packets at the AS boundary.

Because of this lack of data, an in-depth quantitative performance comparison between Puri's thesis and this research is not possible. Even if it is assumed that the 20-second time was obtained under an attack with the maximum bit rate of 16.76 Mbps, that performance is significantly worse than the time achieved by this current network for the same attack flow.

Stamatelatos reported the average response times by a similar network setup albeit using BGP BHR, for a comparable set of attack flows [2]. His definition of response time, however, was based on the events of BGP route advertisements as he stated in his thesis [2]:

> The main performance metric for this research was the response time of the routers. And the most accurate way to measure those values was to capture the trigger router's initial routing-advertisement update and the border routers' subsequent routing update messages.

This definition does not take into account the time that it takes to detect the attack and the time it takes to add a new static route to the trigger router. This was expected because no IDS was used and the static route was added to the trigger router manually in those experiments. Thus, the response times reported in Stamatelatos's thesis [2] might have underestimated the actual response times. In contrast, this research's definition of mitigation time includes all necessary steps from the attack's detection to its mitigation.

Figure 38 shows the average mitigation times from the current experiments together with the response times reported in Figure 36 of Stamatelatos's thesis [2]. Even discounting the possible underestimation factor, this current network responded to the attacks slightly faster. Another factor is worth mentioning. The response times reported in Figure 36 of Stamatelatos's

thesis were obtained using a Juniper router as the core router. That router has a much higher CPU and link capacity than the Cisco router used for LSR1. Therefore, the Juniper router most likely did not suffer from the same anomalies that delayed mitigation by this research's network.

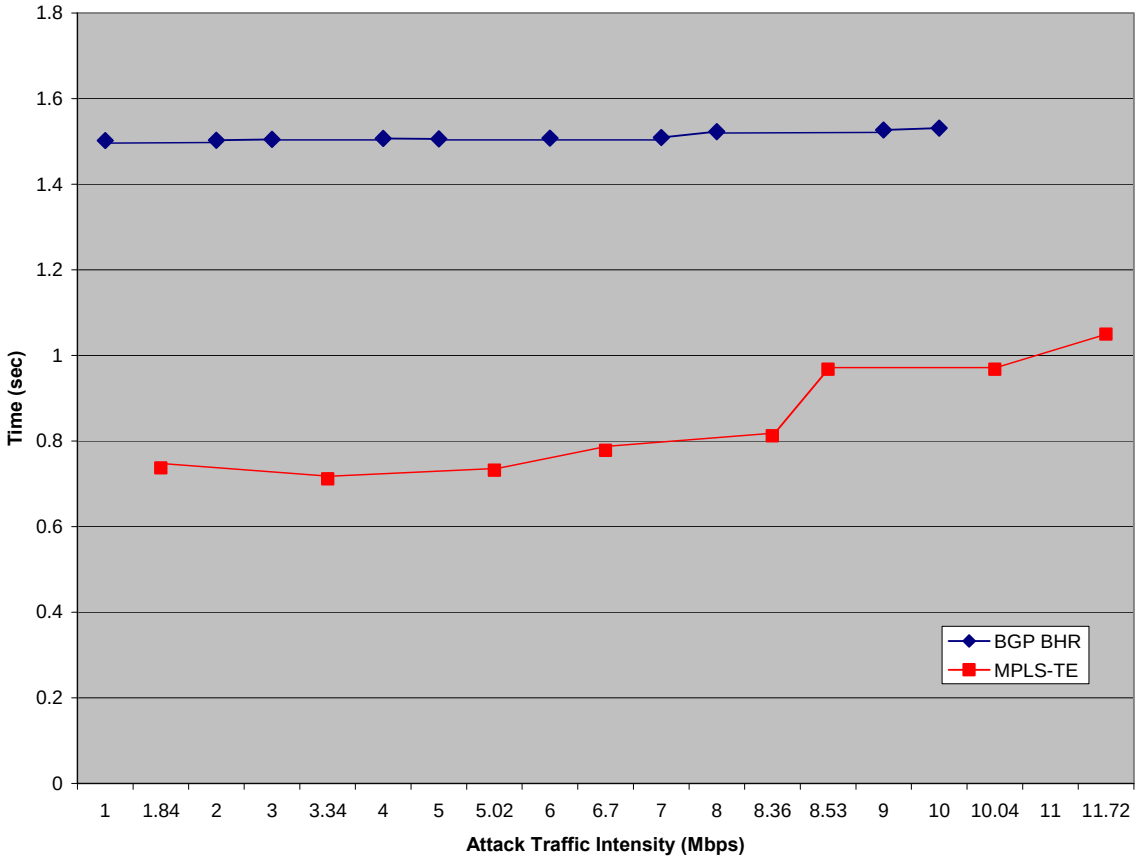


Figure 38. Comparison of MPLS-TE and BGP BHR techniques

The slightly better performance of the MPLS-TE technique may be due to use of the "priority list" command to give telnet sessions priority over other traffic. In contrast, BGP routing messages did not receive such preferential treatment in Stamatelatos' BGP BHR work.

The most significant advantage of the current technique compared to BGP BHR is that it does not indiscriminately drop packets destined to the target host. The traffic could be redirected to the cleaning center where it could be analyzed,

sanitized, and then sent to the target host. The forwarding of sanitized traffic to the target host can be easily accomplished with a slight modification to this network: adding a second LER for the cleaning center and configuring a dedicated tunnel from the new LER to the target host. With this approach no legal packet “will be sent to the trash,” even during an attack.

THIS PAGE INTENTONALLY LEFT BLANK

V CONCLUSIONS

A. CONCLUS IONS

Using a real test-bed network, this study evaluated the performance of the proposed MPLS-TE technique for mitigating DDoS attacks. In Chapter IV, a real-time fully automated attack detection and mitigation process was described. The network was tested under stressful situations in which the router CPU capacity and the link capacity became bottlenecks. In the same chapter the timing results were discussed and compared to the results from two prior studies of the BGP BHR techniques. The above actions led us to the following conclusions.

The MPLS-TE technique provides a relatively simple implementation of the sinkhole routing method. It does not require special interface cards that may add significant processing overhead to the involved routers. The method allowed successful protection of the target host and kept it reachable for legitimate traffic within the AS. The overall system response time can be within seconds; comparable to the best results achieved with BGP BHR. Unlike BGP BHR techniques, the MPLS-TE technique avoids blind discarding of all traffic destined to the target host. Furthermore, it provides the capability to analyze the traffic (malicious or not) for forensics purposes.

The main disadvantage of the MPLS-TE technique is that the infrastructure must be upgraded to support MPLS. This means that routers with older versions of software need to be replaced.

Two other points should be noted, First, the MPLS-TE proposals in the literature use BGP protocol to advertise the redirection routes. In this study telnet was used for this purpose in order to simplify the router configurations and make the system more controllable. This also made it relatively easy to implement selective unblocking on any border router. As presented in Chapter IV, the telnet technique was very efficient and achieved mitigation times under two seconds

even under intense attack traffic. However telnet appears to have security issues. Those issues can be addressed by using Secure Shell protocol (SSH) instead of telnet, with only a small additional processing overhead.

Second, by not discarding packets at border routers, the MPLS-TE technique may keep the target's AS under stress even after the redirection of the attack traffic. For example, in the current test-bed, when the traffic flow rate went above 8.5 Mbps, the CPU load of the core router exceeded the safe threshold of 80%, and in a short amount of time (less than a minute) the connections between this and other routers became unstable. Therefore, it is important for a network to have enough resources to deal with a large amount of malicious traffic when a sinkhole method is employed.

B. FUTURE WORK

Research in the following two areas will provide a more complete evaluation of the utility of MPLS-TE in mitigation of DDoS attacks.

1. The current study did not implement many important functions of the cleaning center. The cleaning center should perform traffic analysis and cleaning, forensics and archiving, and finally selective forwarding of "clean" packets to the target host. The employment of a suitable cleaning center with the above capabilities would provide a complete, integrated, anti-DDoS solution with which to assess the full mitigative benefits of the MPLS-TE technique.

2. The current study used only one type of low to mid-range Cisco routers. Real-world large networks often consist of many types of routers with different capabilities and from different vendors. Thus, another potential area of future study could involve evaluating the degree of interoperability of MPLS-TE in such heterogeneous environments.

APPENDIX A. ROUTERS' CONFIGURATION FILES

Appendix A presents the configuration files for the rest test-bed's network routers LER1, LER3 and LER4.

Configuration of LER1 – Cleaning Center's Router

```
Current configuration : 2289 bytes
!
version 12.2
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname LER1
!
!
ip subnet-zero
!
!
!
ip cef
mpls traffic-eng tunnels
call rsvp-sync
!
!
!
!
!
!
!
!
interface Loopback0
 ip address 192.168.10.5 255.255.255.255
!
interface Tunnel17
 ip unnumbered Loopback0
 tunnel destination 192.168.10.4
 tunnel mode mpls traffic-eng
 tunnel mpls traffic-eng autoroute announce
 tunnel mpls traffic-eng priority 7 7
 tunnel mpls traffic-eng bandwidth 4800
 tunnel mpls traffic-eng path-option 1 explicit name def-LER2
!
interface Tunnel18
 ip unnumbered Loopback0
 tunnel destination 192.168.10.3
 tunnel mode mpls traffic-eng
 tunnel mpls traffic-eng autoroute announce
 tunnel mpls traffic-eng priority 7 7
 tunnel mpls traffic-eng bandwidth 4800
```

```
 tunnel mpls traffic-eng path-option 1 explicit name def-LER3
!
interface Tunnel19
 ip unnumbered Loopback0
 tunnel destination 192.168.10.2
 tunnel mode mpls traffic-eng
 tunnel mpls traffic-eng autoroute announce
 tunnel mpls traffic-eng priority 7 7
 tunnel mpls traffic-eng bandwidth 100
 tunnel mpls traffic-eng path-option 1 explicit name def-LER4
!
interface FastEthernet0/0
 no ip address
 shutdown
 duplex auto
 speed auto
!
interface Ethernet1/0
 ip address 192.168.2.128 255.255.255.0
 half-duplex
!
interface Ethernet1/1
 ip address 192.168.7.128 255.255.255.0
 half-duplex
 mpls traffic-eng tunnels
 tag-switching ip
 priority-group 1
 ip rsvp bandwidth 10000 10000
!
interface Ethernet1/2
 no ip address
 shutdown
 half-duplex
!
interface Ethernet1/3
 no ip address
 shutdown
 half-duplex
!
router ospf 99
 router-id 192.168.10.5
 log-adjacency-changes
 network 192.168.0.0 0.0.255.255 area 0
 mpls traffic-eng router-id Loopback0
 mpls traffic-eng area 0
!
ip classless
no ip http server
!
ip explicit-path name def-LER4 enable
 next-address 192.168.7.1
 next-address 192.168.6.128
!
ip explicit-path name def-LER2 enable
 next-address 192.168.7.1
```

```
 next-address 192.168.4.128
!
ip explicit-path name def-LER3 enable
 next-address 192.168.7.1
 next-address 192.168.5.128
!
priority-list 1 protocol ip high tcp telnet
priority-list 1 protocol ip low
!
!
dial-peer cor custom
!
!
!
!
line con 0
 password vordos
 login
line aux 0
line vty 0 4
 password vordos
 login
!
end
```

Configuration of LER3 – Right Border Router

```
Current configuration : 1916 bytes
!
version 12.2
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname LER3
!
!
ip subnet-zero
!
!
!
ip cef
ip audit notify log
ip audit po max-events 100
mpls traffic-eng tunnels
!
call rsvp-sync
!
!
!
!
!
!
!
```

```
!
interface Loopback0
 ip address 192.168.10.3 255.255.255.255
!
interface Tunnel10
 ip unnumbered Loopback0
 tunnel destination 192.168.10.2
 tunnel mode mpls traffic-eng
 tunnel mpls traffic-eng autoroute announce
 tunnel mpls traffic-eng priority 7 7
 tunnel mpls traffic-eng bandwidth 4800
 tunnel mpls traffic-eng path-option 1 explicit name def-LER4
!
interface Tunnel11
 ip unnumbered Loopback0
 tunnel destination 192.168.10.5
 tunnel mode mpls traffic-eng
 tunnel mpls traffic-eng autoroute announce
 tunnel mpls traffic-eng priority 7 7
 tunnel mpls traffic-eng bandwidth 4800
 tunnel mpls traffic-eng path-option 1 explicit name def-LER1
!
interface Ethernet0/0
 ip address 192.168.8.128 255.255.255.0
 half-duplex
!
interface Ethernet0/1
 description Connection to LSR1
 ip address 192.168.5.128 255.255.255.0
 half-duplex
 mpls traffic-eng tunnels
 tag-switching ip
 priority-group 1
 ip rsvp bandwidth 10000 10000
!
interface Ethernet1/0
 no ip address
 shutdown
 half-duplex
!
interface Ethernet1/1
 ip address 192.168.0.102 255.255.255.252
 half-duplex
!
router ospf 99
 router-id 192.168.10.3
 log-adjacency-changes
 network 192.168.0.0 0.0.255.255 area 0
 mpls traffic-eng router-id Loopback0
 mpls traffic-eng area 0
!
ip classless
no ip http server
!
ip explicit-path name def-LER4 enable
```

```
 next-address 192.168.5.1
 next-address 192.168.6.128
!
ip explicit-path name def-LER1 enable
 next-address 192.168.5.1
 next-address 192.168.7.128
!
priority-list 1 protocol ip high tcp telnet
priority-list 1 protocol ip low
!
!
dial-peer cor custom
!
!
!
!
!
line con 0
 password vordos
line aux 0
line vty 0 4
 password vordos
 login
!
end
```

Configuration of LER4 – Target Network's Router

```
Current configuration : 2339 bytes
!
version 12.2
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname LER4
!
!
ip subnet-zero
!
!
!
ip cef
mpls traffic-eng tunnels
call rsvp-sync
!
!
!
!
!
!
!
!
interface Loopback0
```

```
 ip address 192.168.10.2 255.255.255.255
!
interface Tunnel14
 ip unnumbered Loopback0
 tunnel destination 192.168.10.5
 tunnel mode mpls traffic-eng
 tunnel mpls traffic-eng autoroute announce
 tunnel mpls traffic-eng priority 7 7
 tunnel mpls traffic-eng bandwidth 100
 tunnel mpls traffic-eng path-option 1 explicit name def-LER1
!
interface Tunnel15
 ip unnumbered Loopback0
 tunnel destination 192.168.10.4
 tunnel mode mpls traffic-eng
 tunnel mpls traffic-eng autoroute announce
 tunnel mpls traffic-eng priority 7 7
 tunnel mpls traffic-eng bandwidth 4800
 tunnel mpls traffic-eng path-option 1 explicit name def-LER2
!
interface Tunnel16
 ip unnumbered Loopback0
 tunnel destination 192.168.10.3
 tunnel mode mpls traffic-eng
 tunnel mpls traffic-eng autoroute announce
 tunnel mpls traffic-eng priority 7 7
 tunnel mpls traffic-eng bandwidth 4800
 tunnel mpls traffic-eng path-option 1 explicit name def-LER3
!
interface Ethernet0/0
 ip address 192.168.3.128 255.255.255.0
 half-duplex
!
interface Ethernet0/1
 no ip address
 shutdown
 half-duplex
!
interface Ethernet0/2
 description Connection to LSR1
 ip address 192.168.6.128 255.255.255.0
 half-duplex
 mpls traffic-eng tunnels
 tag-switching ip
 priority-group 1
 ip rsvp bandwidth 10000 10000
!
interface Ethernet0/3
 no ip address
 shutdown
 half-duplex
!
interface FastEthernet1/0
 no ip address
 shutdown
```

```
 duplex auto
 speed auto
!
router ospf 99
 router-id 192.168.10.2
 log-adjacency-changes
 network 192.168.0.0 0.0.255.255 area 0
 mpls traffic-eng router-id Loopback0
 mpls traffic-eng area 0
!
ip classless
no ip http server
!
ip explicit-path name def-LER1 enable
 next-address 192.168.6.1
 next-address 192.168.7.128
!
ip explicit-path name def-LER2 enable
 next-address 192.168.6.1
 next-address 192.168.4.128
!
ip explicit-path name def-LER3 enable
 next-address 192.168.6.1
 next-address 192.168.5.128
!
priority-list 1 protocol ip high tcp telnet
priority-list 1 protocol ip low
!
!
dial-peer cor custom
!
!
!
!
gatekeeper
 shutdown
!
!
line con 0
 password vordos
line aux 0
line vty 0 4
 password vordos
 login
!
end
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B. SSP_CISCO_NULLROUTE.C FILE

Appendix B presents the modified source code for the Cisco null route plug-in. The original code was developed by Mr. Frank Knobbe [29].

```
/* $Id: ssp_cisco_nullroute.c,v 2.3 2008/04/26 19:50:26 fknobbe Exp $
 *
 *
 * Copyright (c) 2005-2008 Frank Knobbe <frank@knobbe.us>
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
the
 *    documentation and/or other materials provided with the
distribution.
 *
 * THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS ``AS IS''
AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE
LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE
GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN
ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY
OF
 * SUCH DAMAGE.
 *
 * Acknowledgements:
 *
 * Brent Erickson and Sergio Salazar for the idea and sample commands.
 *
 *
 * ssp_cisco_nullroute.c
```

```
 *
 * Purpose:
 *
 * This SnortSam plugin telnet's into one or more Cisco routers and
issues
 * a route command to effectively "null-route" the intruding IP
address.
 * SnortSam will remove the added routes when the blocks expire.
 *
 *
 */


#ifndef           __SSP_CISCO_NULLROUTE_C__
#define           __SSP_CISCO_NULLROUTE_C__


#include "snortsam.h"
#include "ssp_cisco_nullroute.h"


#include <sys/types.h>
#include <stdio.h>
#include <string.h>
#include <time.h>
#ifdef WIN32
#include <winsock.h>
#else
#include <netinet/in.h>
#include <arpa/inet.h>
#endif



/* This routine parses the cisconullroute statements in the config
file.
 * It builds a list of routers)
*/
void CiscoNullRouteParse(char *val,char *file,unsigned long
line,DATALIST *plugindatalist)
{     CISCONULLROUTEDATA *ciscop;
      char *p2,msg[STRBUFSIZE+2],*p3;
      struct in_addr routerip;

#ifdef FWSAMDEBUG
      printf("Debug: [cisconullroute] Plugin Parsing...\n");
#endif

      if(*val)
      {     p2=val;
            while(*p2 && !myisspace(*p2))
                  p2++;
            if(*p2)
                  *p2++ =0;
```

```
            routerip.s_addr=getip(val);
            if(routerip.s_addr)                /* If we have a valid
IP address */
            {
      ciscop=safemalloc(sizeof(CISCONULLROUTEDATA),"ciscoparse","ciscop
");   /* create new router */
                  plugindatalist->data=ciscop;
                  ciscop->ip.s_addr=routerip.s_addr;
                  ciscop->routersocket=0;
                  ciscop->loggedin=FALSE;
                  ciscop->username[0]=ciscop->enablepw[0]=ciscop-
>userlogin=0;
                  ciscop->telnetpw=ciscop->username;

                  if(*p2)
                  {     val=p2;
                        while(*val && myisspace(*val))      /* now
parse the remaining text */
                              val++;
                        if(val)
                        {     p2=val;
                              while(*p2 && !myisspace(*p2))
                                    p2++;
                              if(*p2)
                                    *p2++ =0;
                              safecopy(ciscop->username,val);     /*
save telnet password */

                              p3=strchr(ciscop->username,'/');  /*
Check if a username is given */
                              if(p3)
                              {     *p3++ =0;
                                    ciscop->telnetpw=p3;
                                    ciscop->userlogin=TRUE;
                              }

                              if(*p2)
                  /* if we have a second password */
                              {     while(*p2 && myisspace(*p2))
                                          p2++;
                                    safecopy(ciscop->enablepw,p2);/* it
would be the enable password */
                              }
                              else
                                    safecopy(ciscop->enablepw,ciscop-
>telnetpw); /* if only one password was found, use it for both */
                        }
                  }
                  if(!ciscop->telnetpw[0])
                  {     snprintf(msg,sizeof(msg)-1,"Error: [%s: %lu]
Cisco Router defined without passwords!",file,line);
                        logmessage(1,msg,"cisconullroute",0);
                        free(ciscop);
                        plugindatalist->data=NULL;
                  }
```

```
#ifdef FWSAMDEBUG
                  else
                        printf("Debug: [cisconullroute] Adding Cisco
Router: IP \"%s\", PW \"%s\", EN \"%s\"\n",inettoa(ciscop-
>ip.s_addr),ciscop->telnetpw,ciscop->enablepw);
#endif
            }
            else
            {     snprintf(msg,sizeof(msg)-1,"Error: [%s: %lu] Invalid
CiscoNullRoute parameter '%s' ignored.",file,line,val);
                  logmessage(1,msg,"cisconullroute",0);
            }
      }
      else
      {     snprintf(msg,sizeof(msg)-1,"Error: [%s: %lu] Empty
CiscoNullRoute parameter.",file,line);
            logmessage(1,msg,"cisconullroute",0);
      }
}


/* This routine initiates the block. It walks the list of routers
 * telnet's in, and issues the route command.
 */
void CiscoNullRouteBlock(BLOCKINFO *bd,void *data,unsigned long qp)
{   CISCONULLROUTEDATA *ciscop;
      struct sockaddr_in thissocketaddr,routersocketaddr;
      unsigned long flag;
      char cnrmsg[STRBUFSIZE+1],cnrat[STRBUFSIZE+1];
#ifdef FWSAMDEBUG
#ifdef WIN32
      unsigned long threadid=GetCurrentThreadId();
#else
      pthread_t threadid=pthread_self();
#endif
#endif

      if(!data)
            return;
    ciscop=(CISCONULLROUTEDATA *)data;

#ifdef FWSAMDEBUG
      printf("Debug: [cisconullroute][%lx] Plugin
Blocking...\n",(unsigned long)threadid);
#endif

      snprintf(cnrat,sizeof(cnrat)-1,"router at %s",inettoa(ciscop-
>ip.s_addr));

      if(!ciscop->routersocket)
      {     routersocketaddr.sin_port=htons(23); /* telnet */
            routersocketaddr.sin_addr.s_addr=ciscop->ip.s_addr;
            routersocketaddr.sin_family=AF_INET;

            thissocketaddr.sin_port=htons(0); /* get a dynamic port  */
```

```
            thissocketaddr.sin_addr.s_addr=0;
            thissocketaddr.sin_family=AF_INET;

            /* create socket */
            ciscop-
>routersocket=socket(PF_INET,SOCK_STREAM,IPPROTO_TCP);
            if(ciscop->routersocket==INVALID_SOCKET)
            {     snprintf(cnrmsg,sizeof(cnrmsg)-1,"Error:
[cisconullroute] Couldn't create socket!");
                  logmessage(1,cnrmsg,"cisconullroute",ciscop-
>ip.s_addr);
                  ciscop->routersocket=0;
                  return;
            }
            /* bind it */
            if(bind(ciscop->routersocket,(struct sockaddr
*)&(thissocketaddr),sizeof(struct sockaddr)))
            {     snprintf(cnrmsg,sizeof(cnrmsg)-1,"Error:
[cisconullroute] Couldn't bind socket!");
                  logmessage(1,cnrmsg,"ciscocnullroute",ciscop-
>ip.s_addr);
                  ciscop->routersocket=0;
                  return;
            }
            /* and connect to router */
            if(connect(ciscop->routersocket,(struct sockaddr
*)&routersocketaddr,sizeof(struct sockaddr)))
            {     snprintf(cnrmsg,sizeof(cnrmsg)-1,"Error:
[cisconullroute] Could not connect to %s! Will try later.",cnrat);
                  logmessage(1,cnrmsg,"cisconullroute",ciscop-
>ip.s_addr);
                  closesocket(ciscop->routersocket);
                  ciscop->routersocket=0;
            }
      }
      if(ciscop->routersocket)
      {     do
            {
#ifdef FWSAMDEBUG
                  printf("Debug: [cisconullroute][%lx] Connected to
%s.\n",(unsigned long)threadid,cnrat);
#endif
                  flag=-1;
                  ioctlsocket(ciscop->routersocket,FIONBIO,&flag);
      /* set non blocking  */
                  flag=FALSE;

                  if(!ciscop->loggedin)
                  {     if(ciscop->userlogin)
                        {     if(!sendreceive(ciscop-
>routersocket,CNRNETWAIT,"cisconullroute",ciscop-
>ip,"","username","waiting for user logon prompt from ",cnrat))
                              {     flag=TRUE;
                                    continue;
                              }
```

```
                              snprintf(cnrmsg,sizeof(cnrmsg)-
1,"%s\r",ciscop->username);   /* Send username password */

                              if(!sendreceive(ciscop-
>routersocket,CNRNETWAIT,"cisconullroute",ciscop->ip,cnrmsg,"pass","at
password prompt from ",cnrat))
                              {     flag=TRUE;
                                    continue;
                              }
                              snprintf(cnrmsg,sizeof(cnrmsg)-
1,"%s\r",ciscop->telnetpw);   /* Send telnet password */
                        }
                        else
                        {     if(!sendreceive(ciscop-
>routersocket,CNRNETWAIT,"cisconullroute",ciscop->ip,"","pass","waiting
for logon prompt from ",cnrat))
                              {     flag=TRUE;
                                    continue;
                              }
                              snprintf(cnrmsg,sizeof(cnrmsg)-
1,"%s\r",ciscop->telnetpw);   /* Send telnet password */
                        }

                        if(!sendreceive(ciscop-
>routersocket,CNRNETWAIT,"cisconullroute",ciscop->ip,cnrmsg,">","at
logon prompt of ",cnrat))
                        {     flag=TRUE;
                              continue;
                        }

                        /* Send enable */

                        //Changed by the author to the minimum accepted
command ("en" instead of "enable") by the cisco routers to reduce
session's duration.
                        if(!sendreceive(ciscop-
>routersocket,CNRNETWAIT,"cisconullroute",ciscop->ip,"en\r","pass","at
enable command of ",cnrat))
                        {     flag=TRUE;
                              continue;
                        }

                        /* Send enable password */
                        snprintf(cnrmsg,sizeof(cnrmsg)-1,"%s\r",ciscop-
>enablepw);
                        if(!sendreceive(ciscop-
>routersocket,CNRNETWAIT,"cisconullroute",ciscop->ip,cnrmsg,"#","at
enable prompt of ",cnrat))
                        {     flag=TRUE;
                              continue;
                        }

                        /* Send config */
```

```
                        //Changed by the author to the minimum accepted
command ("conf t" instead of "configuration terminal") by the cisco
routers to reduce session's duration.
                        if(!sendreceive(ciscop-
>routersocket,CNRNETWAIT,"cisconullroute",ciscop->ip,"conf t\r","#","at
config command of ",cnrat))
                        {     flag=TRUE;
                              continue;
                        }
                        ciscop->loggedin=TRUE;
                  }

                  /* send route command */
                  // The below 3 "if" commands added by the author in
order to chose the correct static route to be added for the
coresponding router.
                   if(strcmp(inettoa(ciscop->ip.s_addr),
"192.168.10.5") == 0) {
                        // Changed by the author in order to fit to add
to LER1 the correct static route.
                        snprintf(cnrmsg,sizeof(cnrmsg)-1,"%sip route %s
255.255.255.255 e1/0\r",bd->block?"":"no ",inettoa(bd->blockip));
                  }

                  if(strcmp(inettoa(ciscop->ip.s_addr), "192.168.10.3")
== 0) {
                        // Changed by the author in order to fit to add
to LER3 the correct static route.
                        snprintf(cnrmsg,sizeof(cnrmsg)-1,"%sip route %s
255.255.255.255 t11\r",bd->block?"":"no ",inettoa(bd->blockip));
                  }

                  .
                  if(strcmp(inettoa(ciscop->ip.s_addr), "192.168.10.4")
== 0) {
                        // Changed by the author in order to fit to add
to LER1 the correct static route
                        snprintf(cnrmsg,sizeof(cnrmsg)-1,"%sip route %s
255.255.255.255 t12\r",bd->block?"":"no ",inettoa(bd->blockip));
                  }
                  if(!sendreceive(ciscop-
>routersocket,CNRNETWAIT,"cisconullroute",ciscop->ip,cnrmsg,"#","at
route command of ",cnrat))
                  {     flag=TRUE;
                        continue;
                  }

                  if(!moreinqueue(qp))
                  {     /* End input */
                        if(!sendreceive(ciscop-
>routersocket,CNRNETWAIT,"cisconullroute",ciscop->ip,"\032","#","at
CTRL-Z of ",cnrat))
                        {     flag=TRUE;
                              continue;
                        }
```

```
                        //* Save config */
                        // Ignored by the author in order to reduse
each session's duration. We don't need permanently store the static
routes.
                        //if(!sendreceive(ciscop-
>routersocket,CNRNETWAIT,"cisconullroute",ciscop->ip,"write
mem\r","#","at write mem command of ",cnrat))
                        //{   flag=TRUE;
                        //    continue;
                        //}

                        /* and we're outta here... */
                        sendreceive(ciscop-
>routersocket,CNRNETWAIT,"cisconullroute",ciscop->ip,"quit\r","","at
quit command of ",cnrat);
                        flag=TRUE;
                        //Changed by the author to FALSE in order to
avoid continously keeping telnet session open.
                        ciscop->loggedin = FALSE;
                  }
            }while(FALSE);

            if(flag)
            {     closesocket(ciscop->routersocket);
                  ciscop->routersocket=0;
                  ciscop->routersocket=FALSE;
            }
      }
}

#endif /* __SSP_CISCO_NULLROUTE_C__ */
```

LIST OF REFERENCES

[1] G. C. Kessler, “Defenses Against Distributed Denial of Service Attacks”, November 2000, http://www.garykessler.net/library/ddos.html, Retrieved 10/30/2008.

[2] N. Stamatelatos, “A Measurement Study of BGP Black Hole Routing Performance,” M.S. Thesis, Naval Postgraduate School, CA, USA, September 2006.

[3] V. Puri, “Automated Alerting for Black Hole Routing”, M.S. Thesis, Naval Postgraduate School, CA, USA, September 2007.

[4] World Wide Web Consortium, “Securing against Denial of Service attacks”, February 2003, http://www.w3.org/Security/faq/wwwsf6.html#DOS-Q2, Retrieved 11/20/2008.

[5] S. M. Specht and R. B. Lee, “Distributed denial of service: taxonomies of attacks, tools and countermeasures," in Proceedings of the 17th International Conference on Parallel and Distributed Computing Systems, San Francisco, Sept. 15-17, 2004, pp. 543-550.

[6] Cs3 Inc., “What is DDoS and How It is Hurting the Internet, E-commerce and Business”, http://www.cs3-inc.com/pk_whatisddos.html, Retrieved 01/19/09.

[7] J.E.Canavan, “Fundamentals of Network Security.” Norwood: Artech House Inc., 2001.

[8] Department of Computer Science and Information Engineering at National Central University of Taiwan, “What is Push+ACK Attack?” http://www.csie.ncu.edu.tw/~cs102085/DDoS/protocolexploit/push%2Back/description.htm, Retrieved 11/22/2008.

[9] E. Skoudis and T. Liston, *Counter Hack Reloaded, Second Edition: A Step-by-Step Guide to Computer Attacks and Effective Defenses*. Massachussets: Pearson Education Inc., 2005.

[10] International Engineering Consortium, “Multiprotocol Label Switching (MPLS) Definition and Overview,” 2007, http://www.iec.org/online/tutorials/mpls/, Retrieved 11/23/2008.

[11] Cisco Systems Inc., “MPLS/Tag Switching,” 1992-2009, http://www.cisco.com/en/US/docs/internetworking/technology/handbook/MPLS_Tag-Switching.html, Retrieved 01/5/2009.

[12] C.Mendiratta, “Selecting an MPLS provider,“ 2001, http://www.expresscomputeronline.com/20070702/technology02.shtml, Retrieved 01/8/2009.

[13] Wikipedia, “Multiprotocol Label Switching,” http://en.wikipedia.org/wiki/Multiprotocol_Label_Switching, 2009, Retrieved 01/16/2009.

[14] G. W. Cox, “Label Switching”, Spring 2004, http://www.cs.uah.edu/~gcox/670docs/mpls.pdf, Retrieved 08/07/2008.

[15] M. Ivcovic, “MPLS”, 2004, http://www.it.lut.fi/kurssit/07-08/CT30A2300/Verkkomateriaali/Day3/Day3-Luento19_MPLS-2.pdf, Retrieved 08/07/2008.

[16] Juniper Networks Inc., ”How MPLS Works,” 1998-2009, http://www.juniper.net/techpubs/software/erx/junose72/swconfig-bgp-mpls/html/mpls-config5.html, Retrieved 01/9/2009.

[17] W. Riedel,” MPLS Overview,” 2001, http://pagesperso-orange.fr/wallu/MPLS-overview.pdf, Retrieved 01/16/2009.

[18] Protocols.com, “MPLS,” http://www.protocols.com/pbook/mpls.htm, Retrieved 01/17/09.

[19] Wikipedia, “Teletraffic engineering,” http://en.wikipedia.org/wiki/Traffic_engineering_(telecommunications), Retrieved 06/16/08.

[20] Cisco Systems Inc., “Multiprotocol Label Switching (MPLS) Traffic Engineering,” 1992-2009, http://www.cisco.com/univercd/cc/td/doc/product/software/ios120/120newft/120limit/120s/120s5/mpls_te.htm#wp37345, Retrieved 01/17/2009

[21] Y. Afek , R. Brooks and N. Fischbach, “MPLS-based traffic shunt,” presented in twenty eighth North American Network Operator’s Group meeting, Salt Lake City, Utah, June 2003.

[22] N. Fischbach, “MPLS-based traffic shunt,” presented in forty sixth Réseaux IP Européens Network Coordination Center’s (RIPE NCC) meeting, Amsterdam, Netherlands, September 2003.

[23] Huawei Technologies Co. Ltd., “Technical White Paper for Sinkhole routing,” 2007, http://www.huawei.com/products/datacomm/pdf/view.do?f=83, Retrieved 01/18/2009.

[24] Cisco Systems Inc., “MPLS Basic Traffic Engineering Using OSPF Configuration Example,” 1992-2009, http://www.cisco.com/en/US/tech/tk436/tk428/technologies_configuration_example09186a0080093fd0.shtml, Retrieved 01/23/2009.

[25] Cisco Systems Inc., “Understanding and Configuring the ip unnumbered Command,” 1992-2009, http://www.cisco.com/en/US/tech/tk648/tk362/technologies_tech_note09186a0080094e8d.shtml, Retrieved 01/23/2009.

[26] P. Harper, “Snort Enterprise Install: Snort, Apache, SSL, PHP, MySQL, and BASE Install on CentOS 4, RHEL 4 or Fedora Core – with NTOP,” 2006, http://www.snort.org/docs/setup_guides/Snort_Base_Minimal.pdf, Retrieved 07/20/08.

[27] M.Jonkmann, “SnortSam Install,” 2007, http://doc.emergingthreats.net/bin/view/Main/SnortSamINSTALL, Retrieved 9/5/08.

[28] Cisco Systems Inc., “OSPF neighbor flaps and receives the OSPF-5-ADJCHG error message,” 2008, http://supportwiki.cisco.com/ViewWiki/index.php/OSPF_neighbor_flaps_and_receives_the_OSPF-5-ADJCHG_error_message, Retrieved 01/28/09.

[29] SnortSam.net, “SnortSam Download,” 2001-2007, http://www.snortsam.net/download.html, Retrieved 01/23/09.

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, VA

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, CA

3. Geoffrey Xie
   Naval Postgraduate School
   Monterey, CA

4. John D. Fulp
   Naval Postgraduate School
   Monterey, CA

5. Ioannis Vordos
   Naval Postgraduate School
   Monterey, CA

6. Neal Ziring
   National Security Agency
   Fort Meade, MD

7. Trent Pitsenbarger
   National Security Agency
   Fort Meade, MD
   t.pitsen@radium.ncsc.mil

8. Terry Dossey
   National Security Agency
   Fort Meade, MD